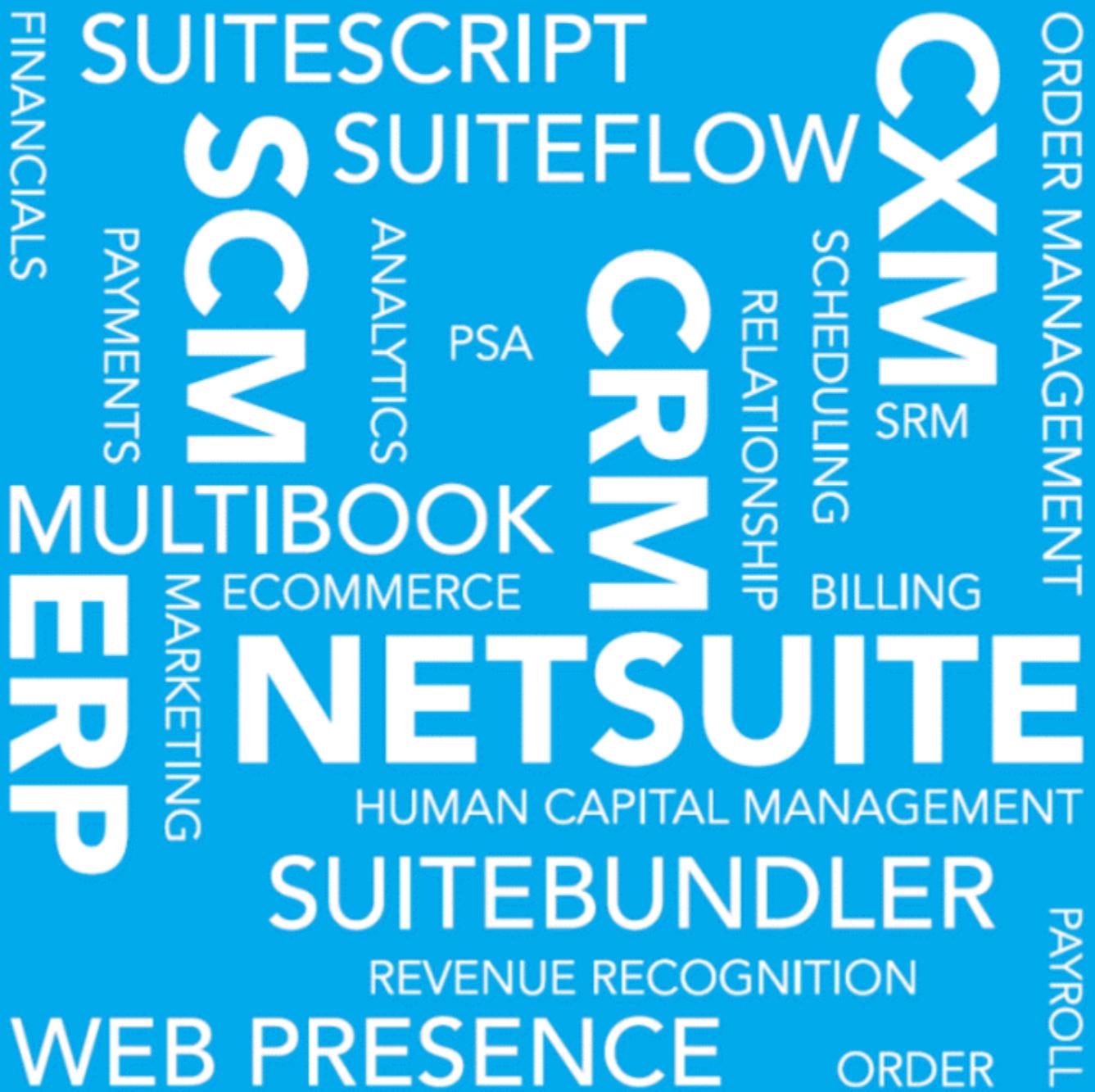


# SuiteScript 2.0 API Reference



Copyright © 2005, 2019, Oracle and/or its affiliates. All rights reserved.

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

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

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc.

AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

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

This documentation is in pre-General Availability status and is intended for demonstration and preliminary use only. It may not be specific to the hardware on which you are using the software. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to this documentation and will not be responsible for any loss, costs, or damages incurred due to the use of this documentation.

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

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your pre-General Availability trial agreement only. It is not a commitment to deliver any material, code, or functionality, and

should not be relied upon in making purchasing decisions. The development, release, 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](http://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](mailto:support@netsuite.com) or other designated contact for the specific beta feature. Oracle cannot guarantee the continued availability of such beta features and may substantially modify or cease providing such beta features without entitling Customer to any refund, credit, or other compensation. Oracle makes no representations or warranties regarding functionality or use of beta features and Oracle shall have no liability for any lost data, incomplete data, re-run time, inaccurate input, work delay, lost profits or adverse effect on the performance of the Service resulting from the use of beta features. Oracle's standard service levels, warranties and related commitments regarding the Service shall not apply to beta features and they may not be fully supported by Oracle's customer support. These limitations and exclusions shall apply until the date that Oracle at its sole option makes a beta feature generally available to its customers and partners as part of the Service without a "beta" label.

# Send Us Your Feedback

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

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

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

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

To report software issues, contact NetSuite Customer Support.

# Table of Contents

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) .....	11
SuiteScript 2.0 Global Objects .....	36
define Object .....	36
define(moduleObject) .....	37
define(id, [dependencies], moduleObject) .....	38
require Function .....	41
require([dependencies], callback) .....	41
require Configuration .....	43
log Object .....	44
util Object .....	45
toString() .....	45
JSON object .....	46
JSON.parse(text) .....	46
JSON.stringify(obj) .....	47
Promise Object .....	48
SuiteScript 2.0 Modules .....	51
N/action Module .....	53
action.Action .....	59
action.execute(options) .....	70
action.execute.promise(options) .....	71
action.executeBulk(options) .....	72
action.find(options) .....	74
action.find.promise(options) .....	75
action.get(options) .....	77
action.get.promise(options) .....	78
N/auth Module .....	79
auth.changeEmail(options) .....	80
auth.changePassword(options) .....	81
N/cache Module .....	82
cache.Cache .....	86
cache.getCache(options) .....	91
cache.Scope .....	92
N/certificateControl Module .....	93
certificateControl.Certificate .....	99
certificateControl.createCertificate(options) .....	109
certificateControl.findCertificates(options) .....	110
certificateControl.findUsages(options) .....	111
certificateControl.deleteCertificate(options) .....	112
certificateControl.loadCertificate(options) .....	113
certificateControl.Operation .....	114
certificateControl.Operator .....	115
certificateControl.Type .....	116
N/commerce Modules .....	117
N/commerce/recordView Module .....	117
N/config Module .....	122
config.load(options) .....	123
config.Type .....	125
N/crypto Module .....	126
crypto.Cipher .....	131
crypto.CipherPayload .....	133
crypto.Decipher .....	135

crypto.Hash .....	138
crypto.Hmac .....	140
crypto.SecretKey .....	142
crypto.createCipher(options) .....	144
crypto.createDecipher(options) .....	145
crypto.createHash(options) .....	146
crypto.createHmac(options) .....	147
crypto.createSecretKey(options) .....	147
crypto.EncryptionAlg .....	148
crypto.HashAlg .....	149
crypto.Padding .....	150
N/crypto/certificate Module .....	151
certificate.SignedXml .....	154
certificate.Signer .....	156
certificate.Verifier .....	158
certificate.createSigner(options) .....	160
certificate.createVerifier(options) .....	161
certificate.verifyXmlSignature(options) .....	161
certificate.signXml(options) .....	162
certificate.HashAlg .....	163
N/currency Module .....	163
currency.exchangeRate(options) .....	164
N/currentRecord Module .....	166
currentRecord.Column .....	174
currentRecord.CurrentRecord .....	177
currentRecord.Field .....	223
currentRecord.Sublist .....	233
currentRecord.get() .....	237
currentRecord.get.promise() .....	238
N/email Module .....	238
email.send(options) .....	240
email.send.promise(options) .....	244
email.sendBulk(options) .....	244
email.sendBulk.promise(options) .....	248
email.sendCampaignEvent(options) .....	248
email.sendCampaignEvent.promise(options) .....	250
N/encode Module .....	251
encode.convert(options) .....	252
encode.Encoding .....	253
N/error Module .....	254
error.SuiteScriptError .....	257
error.UserEventError .....	260
error.create(options) .....	264
N/file Module .....	265
file.File .....	272
file.create(options) .....	288
file.delete(options) .....	290
file.load(options) .....	291
file.Encoding .....	293
file.Type .....	294
file.Reader .....	295
N/format Module .....	297
format.format(options) .....	300
format.parse(options) .....	302
format.Type .....	303

format.Timezone .....	305
N/format/i18n Module .....	309
format.CurrencyFormatter .....	313
format.NumberFormatter .....	316
format.spellOut(options) .....	320
format.getCurrencyFormatter(options) .....	320
format.getNumberFormatter(options) .....	321
format.NegativeNumberFormat .....	322
format.Currency .....	323
N/http Module .....	323
http.ClientResponse .....	328
http.ServerRequest .....	331
http.ServerResponse .....	339
http.get(options) .....	351
http.get.promise(options) .....	352
http.delete(options) .....	353
http.delete.promise(options) .....	355
http.post(options) .....	356
http.post.promise(options) .....	357
http.put(options) .....	358
http.put.promise(options) .....	359
http.request(options) .....	360
http.request.promise(options) .....	362
http.CacheDuration .....	363
http.Method .....	364
http.RedirectType .....	364
N/https Module .....	365
https.SecureString .....	372
https.createSecureKey(options) .....	376
https.createSecureKey.promise(options) .....	377
https.createSecureString(options) .....	378
https.createSecureString.promise(options) .....	379
https.ClientResponse .....	380
https.ServerRequest .....	383
https.ServerResponse .....	390
https.get(options) .....	401
https.get.promise(options) .....	402
https.delete(options) .....	403
https.delete.promise(options) .....	405
https.post(options) .....	406
https.post.promise(options) .....	407
https.put(options) .....	408
https.put.promise(options) .....	409
https.request(options) .....	410
https.request.promise(options) .....	412
https.CacheDuration .....	413
https.Encoding .....	413
https.HashAlg .....	414
https.Method .....	415
https.RedirectType .....	416
N/https/clientCertificate Module .....	417
clientCertificate.post(options) .....	418
clientCertificate.get(options) .....	419
clientCertificate.put(options) .....	419
clientCertificate.delete(options) .....	420

clientCertificate.request(options) .....	421
N/keyControl Module .....	422
keyControl.Key .....	424
keyControl.createKey(options) .....	429
keyControl.findKeys(options) .....	430
keyControl.deleteKey(options) .....	431
keyControl.loadKey(options) .....	432
keyControl.Operator .....	433
N/log Module .....	434
log.audit(options) .....	436
log.debug(options) .....	437
log.emergency(options) .....	438
log.error(options) .....	439
N/piremoval Module .....	440
piremoval.PiRemovalTask .....	444
piremoval.PiRemovalTaskStatus .....	453
piremoval.PiRemovalTaskLogItem .....	456
piremoval.createTask(options) .....	461
piremoval.deleteTask(options) .....	462
piremoval.getTaskStatus(options) .....	463
piremoval.loadTask(options) .....	464
N/plugin Module .....	466
plugin.findImplementations(options) .....	467
plugin.loadImplementation(options) .....	468
N/portlet Module .....	469
portlet.resize .....	471
portlet.refresh .....	472
N/query Module .....	472
Scripting with the N/query Module .....	489
Formulas in the N/query Module .....	494
Relative Dates in the N/query Module .....	496
SuiteQL in the N/query Module .....	498
query.Column .....	504
query.Component .....	512
query.Condition .....	534
query.Page .....	542
query.PagedData .....	548
query.PageRange .....	553
query.Query .....	556
query.RelativeDate .....	591
query.Result .....	596
query.ResultSet .....	599
query.Sort .....	605
query.SuiteQL .....	612
query.create(options) .....	617
query.createRelativeDate(options) .....	619
query.delete(options) .....	620
query.load(options) .....	621
query.load.promise(options) .....	623
query.runSuiteQL(options) .....	624
query.runSuiteQLPaged(options) .....	625
query.Aggregate .....	627
query.DateId .....	628
query.FieldContext .....	630
query.Operator .....	631

query.RelativeDateRange .....	633
query.ReturnType .....	639
query.SortLocale .....	640
query.Type .....	646
N/record Module .....	656
record.Column .....	678
record.Field .....	682
record.Macro .....	689
record.Record .....	695
record.Sublist .....	761
record.attach(options) .....	766
record.attach.promise(options) .....	768
record.copy(options) .....	770
record.copy.promise(options) .....	771
record.create(options) .....	774
record.create.promise(options) .....	776
record.delete(options) .....	778
record.delete.promise(options) .....	779
record.detach(options) .....	781
record.detach.promise(options) .....	783
record.load(options) .....	784
record.load.promise(options) .....	786
record.submitFields(options) .....	788
record.submitFields.promise(options) .....	791
record.transform(options) .....	793
record.transform.promise(options) .....	796
record.Type .....	799
N/redirect Module .....	801
redirect.redirect(options) .....	803
redirect.toRecord(options) .....	804
redirect.toSavedSearch(options) .....	805
redirect.toSavedSearchResult(options) .....	806
redirect.toSearch(options) .....	807
redirect.toSearchResult(options) .....	808
redirect.toSuitelet(options) .....	808
redirect.toTaskLink(options) .....	809
N/render Module .....	810
render.EmailMergeResult .....	815
render.TemplateRenderer .....	817
render.bom(options) .....	828
render.create() .....	829
render.mergeEmail(options) .....	829
render.packingSlip(options) .....	831
render.pickingTicket(options) .....	832
render.statement(options) .....	833
render.transaction(options) .....	834
render.xmlToPdf(options) .....	835
render.DataSource .....	836
render.PrintMode .....	837
N/runtime Module .....	838
runtime.Script .....	842
runtime.Session .....	848
runtime.User .....	851
runtime.getCurrentScript() .....	859
runtime.getCurrentSession() .....	860

runtime.getCurrentUser()	860
runtime.isFeatureInEffect(options)	861
runtime.accountId	862
runtime.envType	862
runtime.executionContext	863
runtime.processorCount	863
runtime.queueCount	864
runtime.version	865
runtime.ContextType	866
runtime.EnvType	867
runtime.Permission	868
N/search Module	868
search.Search	880
search.Result	894
search.Column	902
search.Filter	911
search.ResultSet	915
search.Page	921
search.PagedData	927
search.PageRange	932
search.Settings	934
search.create(options)	937
search.create.promise(options)	940
search.createSetting(options)	941
search.load(options)	943
search.load.promise(options)	945
search.delete(options)	946
search.delete.promise(options)	947
search.duplicates(options)	948
search.duplicates.promise(options)	950
search.global(options)	951
search.global.promise(options)	952
search.lookupFields(options)	953
search.lookupFields.promise(options)	955
search.createColumn(options)	956
search.createFilter(options)	957
search.Operator	959
search.Sort	960
search.Summary	961
search.Type	962
N/sftp Module	964
Setting up an SFTP Transfer	971
SFTP Authentication	972
Supported Cipher Suites and Host Key Types	975
Supported SuiteScript File Types	975
sftp.Connection	977
sftp.createConnection(options)	987
sftp.MAX_CONNECT_TIMEOUT	990
sftp.MIN_CONNECT_TIMEOUT	990
sftp.MAX_PORT_NUMBER	991
sftp.MIN_PORT_NUMBER	991
sftp.DEFAULT_PORT_NUMBER	992
sftp.Sort	992
N/sso Module	993
sso.generateSuiteSignOnToken(options)	996

N/task Module .....	998
task.ScheduledScriptTask .....	1014
task.ScheduledScriptTaskStatus .....	1018
task.MapReduceScriptTask .....	1020
task.MapReduceScriptTaskStatus .....	1024
task.CsvImportTask .....	1036
task.CsvImportTaskStatus .....	1041
task.EntityDeduplicationTask .....	1042
task.EntityDeduplicationTaskStatus .....	1047
task.SearchTask .....	1048
task.SearchTaskStatus .....	1056
task.WorkflowTriggerTask .....	1059
task.WorkflowTriggerTaskStatus .....	1063
task.RecordActionTask .....	1064
task.RecordActionTaskStatus .....	1071
task.ActionCondition .....	1080
task.create(options) .....	1081
task.checkStatus(options) .....	1085
task.TaskType .....	1086
task.TaskStatus .....	1087
task.MasterSelectionMode .....	1088
task.DedupeMode .....	1089
task.DedupeEntityType .....	1089
task.MapReduceStage .....	1090
N/task/accounting/recognition Module .....	1091
recognition.MergeArrangementsTask .....	1098
recognition.MergeArrangementsTaskStatus .....	1103
recognition.MergeElementsTask .....	1108
recognition.create(options) .....	1112
recognition.checkStatus(options) .....	1113
recognition.TaskStatus .....	1113
recognition.TaskType .....	1114
N/transaction Module .....	1115
transaction.void(options) .....	1117
transaction.void.promise(options) .....	1118
transaction.Type .....	1119
N/translation Module .....	1122
translation.Handle .....	1128
translation.Translator .....	1129
translation.get(options) .....	1130
translation.load(options) .....	1132
translation.selectLocale(options) .....	1134
translation.Locale .....	1135
N/ui Modules .....	1138
N/ui/dialog Module .....	1138
N/ui/message Module .....	1147
N/ui/serverWidget Module .....	1153
N/url Module .....	1295
url.format(options) .....	1298
url.resolveDomain(options) .....	1299
url.resolveRecord(options) .....	1300
url.resolveScript(options) .....	1301
url.resolveTaskLink(options) .....	1302
url.HostType .....	1303
N/util Module .....	1304

util.isArray(obj)	1306
util.isBoolean(obj)	1307
util.isDate(obj)	1308
utilisFunction(obj)	1309
util.isNumber(obj)	1309
utilisObject(obj)	1310
utilisRegExp(obj)	1311
utilisString(obj)	1312
util.nanoTime()	1312
util.each(iterable, callback)	1313
util.extend(receiver, contributor)	1314
N/workflow Module	1315
workflow.initiate(options)	1317
workflow.trigger(options)	1318
N/xml Module	1319
xml.Parser	1329
xml.XPath	1331
xml.Node	1332
xml.Document	1352
xml.Element	1368
xml.Attr	1382
xml.escape(options)	1385
xml.validate(options)	1385
xml.NodeType	1387

# 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 \(nlobj\)](#).

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
<a href="#">nlapiAddDays(d, days)</a>	See Notes	See Notes	<p>This API does not have a SuiteScript 2.0 equivalent.</p> <p>Use the following JavaScript to add or subtract days from a Date object: dateObj.setDate(dateObj.getDate() + or - days)</p> <p>For example:</p> <pre>var tomorrow = new Date(); tomorrow.setDate(tomorrow.getDate() + 1);</pre>

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
			<p><b>Note:</b> SuiteScript 2.0 is also compatible with third party JavaScript APIs that provide this functionality (for example, <a href="#">Moment.js</a>). For information on using third party APIs with SuiteScript 2.0, see the help topic <a href="#">SuiteScript 2.0 Custom Modules</a>.</p>
<a href="#">nlapiAddMonths(d, months)</a>	See Notes	See Notes	<p>This API does not have a SuiteScript 2.0 equivalent.</p> <p>Use the following JavaScript to add or subtract months from a Date object: dateObj.setMonth(dateObj.getMonth() + or - months)</p> <p>For example:</p> <pre>var today = new Date(); var oneMonthAgo = today.setMonth(today.getMonth() - 1);</pre> <p><b>Note:</b> SuiteScript 2.0 is also compatible with third party JavaScript APIs that provide this functionality (for example, <a href="#">Moment.js</a>). For information on using third party APIs with SuiteScript 2.0, see the help topic <a href="#">SuiteScript 2.0 Custom Modules</a>.</p>
<a href="#">nlapiAttachRecord(type, id, type2, id2, attributes)</a>	<code>record.attach(options)</code>	N/record Module	<pre>var recordId = record.attach({   record: {     type: record.Type.FILE,     id: '447'   },   to: {     type: record.type.CUSTOMER,     id: 530   } });</pre>
<a href="#">nlapiCancelLineItem(type)</a>	<code>Record.cancelLine(options)</code> <code>CurrentRecord.cancelLine(options)</code>	N/record Module N/currentRecord Module	-
<a href="#">nlapiCommitLineItem(type)</a>	<code>Record.commitLine(options)</code> <code>CurrentRecord.commitLine(options)</code>	N/record Module N/currentRecord Module	<p>For N/record script samples, see:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">N/record Module Script Samples</a></li> <li>▪ <a href="#">Example: Creating an Inventory Detail Sublist Subrecord</a></li> </ul>
<a href="#">nlapiCopyRecord(type, id, initializeValues)</a>	<code>record.copy(options)</code>	N/record Module	<pre>var recObj = record.copy({   type: record.Type.SALES_ORDER,   id: 284,   isDynamic: true,   defaultValues: {     entity: 547   } }); var recordId = recObj.save();</pre>
<a href="#">nlapiCreateAssistant(title, hideHeader)</a>	<code>serverWidget.createAssistant(options)</code>	N/ui/serverWidget Module	-
<a href="#">nlapiCreateCSVImport()</a>	<code>task.create(options)</code>	N/task Module	For script samples, see <a href="#">N/task Module</a> .

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlapiCreateCurrentLineItemSubrecord(sublist, fldname)	Record.getCurrentSublistSubrecord(options) CurrentRecord.getCurrentSublistSubrecord(options)	N/record Module N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .
nlapiCreateEmailMerger(templateId)	render.mergeEmail(options)	N/render Module	-
nlapiCreateError(code, details, suppressNotification)	error.create(options)	N/error Module	For a script sample, see <a href="#">N/error Module Script Samples</a> .
nlapiCreateFile(name, type, contents)	file.create(options)	N/file Module	For a script sample, see <a href="#">N/file Module Script Samples</a> .
nlapiCreateForm(title, hideNavbar)	serverWidget.createForm(options)	N/ui/serverWidget Module	For a script sample, see <a href="#">N/ui/serverWidget Module Script Samples</a>
nlapiCreateList(title, hideNavbar)	serverWidget.createList(options)	N/ui/serverWidget Module	-
nlapiCreateRecord(type, initializeValues)	record.create(options)	N/record Module	-
nlapiCreateSearch(type, filters, columns)	search.create(options)	N/search Module	For a script sample, see <a href="#">N/search Module Script Samples</a> .
nlapiCreateSubrecord(fldname)	Record.getSubrecord(options) CurrentRecord.getSubrecord(options)	N/record Module N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .
nlapiCreateTemplateRenderer()	render.create()	N/render Module	For a script sample, see <a href="#">N/render Module Script Sample</a> .
nlapiDateToString(d, format)	format.format(options)	N/format Module	For a script sample, see <a href="#">N/format Module Script Samples</a>
nlapiDeleteFile(id)	file.delete(options)	N/file Module	-
nlapiDeleteRecord(type, id, initializeValues)	record.delete(options)	N/record Module	-
nlapiDetachRecord(type, id, type2, id2, attributes)	record.detach(options)	N/record Module	-
nlapiDisableField(fldnam, val)	Field.isDisabled	N/currentRecord Module	Note that <b>isDisabled</b> is a property.
nlapiDisableLineItemField(type, fldnam, val)	Field.isDisabled	N/currentRecord Module	Note that <b>isDisabled</b> is a property.
nlapiEditCurrentLineItemSubrecord(sublist, fldname)	Record.getCurrentSublistSubrecord(options) CurrentRecord.getCurrentSublistSubrecord(options)	N/record Module N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .
nlapiEditSubrecord(fldname)	Record.getSubrecord(options) CurrentRecord.getSubrecord(options)	N/record Module N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .
nlapiEncrypt(s, algorithm, key)	See Notes	See Notes	For SuiteScript 2.0 encryption, hashing, and HMAC functionality, see the <a href="#">N/crypto Module</a> module.  For SuiteScript 2.0 encoding functionality, see the <a href="#">N/encode Module</a> module.
nlapiEscapeXML(text)	xml.escape(options)	N/xml Module	-

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlapiExchangeRate(sourceCurrency, targetCurrency, effectiveDate)	currency.exchangeRate(options)	N/currency Module	For a script sample, see <a href="#">N/currency Module Script Samples</a> .
nlapiFindLineItemMatrixValue(type, fldnam, val, column)	Record.findMatrix SublistLineWithValue(options)  CurrentRecord.findMatrixSublistLineWithValue(options)	N/record Module N/currentRecord Module	-
nlapiFindLineItemValue(type, fldnam, val)	Record.findSublistLineWithValue(options)  CurrentRecord.findSublistLineWithValue(options)	N/record Module N/currentRecord Module	-
nlapiFormatCurrency(str)	format.format(options)	N/format Module	Note that SuiteScript 2.0 currency formatting is handled by the <b>N/format</b> module and not the <b>N/currency</b> module.  For a script sample, see <a href="#">N/format Module Script Samples</a>
nlapiGetContext()	runtime.getCurrentScript()  runtime.getCurrentSession()  runtime.getCurrentUser()	N/runtime Module	For a script sample, see <a href="#">N/runtime Module Script Samples</a> .
nlapiGetCurrentLineItemDateTimeValue(type, fieldId, timeZone)	See Notes	N/format Module	Use the <b>N/format</b> module to mimic this functionality in SuiteScript 2.0.
nlapiGetCurrentLineItemIndex(type)	Record.getCurrentSublistIndex(options)  CurrentRecord.getCurrentSublistIndex(options)	N/record Module N/currentRecord Module	-
nlapiGetCurrentLineItemMatrixValue(type, fldnam, column)	CurrentRecord.getCurrentMatrixSublistValue(options)  Record.getCurrentMatrixSublistValue(options)	N/record Module N/currentRecord Module	-
nlapiGetCurrentLineItemText(type, fldnam)	Record.getCurrentSublistText(options)  CurrentRecord.getCurrentSublistText(options)	N/record Module N/currentRecord Module	-
nlapiGetCurrentLineItemValue(type, fldnam)	Record.getCurrentSublistValue(options)  CurrentRecord.getCurrentSublistValue(options)	N/record Module N/currentRecord Module	-
nlapiGetCurrentLineItemValues(type, fldnam)	Record.getCurrentSublistValue(options)  CurrentRecord.getCurrentSublistValue(options)	N/record Module N/currentRecord Module	-
nlapiGetDateTimeValue(fieldId, timeZone)	See Notes	N/format Module	Use the <b>N/format</b> module to mimic this functionality in SuiteScript 2.0.
nlapiGetDepartment()	User.department	N/runtime Module	-

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlapiGetField(fldnam)	Record.getField (options)  CurrentRecord.getField (options)	N/record Module  N/currentRecord Module	-
nlapiGetFieldText(fldnam)	Record.getText (options)  CurrentRecord.getText (options)	N/record Module  N/currentRecord Module	-
nlapiGetFieldTexts(fldnam)	Record.getText (options)  CurrentRecord.getText (options)	N/record Module  N/currentRecord Module	-
nlapiGetFieldValue(fldnam)	Record.getValue (options)  CurrentRecord. getValue(options)	N/record Module  N/currentRecord Module	-
nlapiGetFieldValues(fldnam)	Record.getValue (options)  CurrentRecord. getValue(options)	N/record Module  N/currentRecord Module	-
nlapiGetJobManager(jobType)	task.create(options)	N/task Module	For a script sample, see <a href="#">N/task Module</a> .
nlapiGetLineItemCount(type)	Record.getLineCount (options)  CurrentRecord. getLineCount(options)	N/record Module  N/currentRecord Module	-
nlapiGetLineItemDateValue(type, fieldId, lineNum, timeZone)	See Notes	N/format Module	Use the <a href="#">N/format</a> module to mimic this functionality in SuiteScript 2.0.
nlapiGetLineItemField(type, fldnam, linenum)	Record.getSublistField (options)  CurrentRecord. getSublistField(options)	N/record Module  N/currentRecord Module	SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.
nlapiGetLineItemMatrixField(type, fldnam, linenum, column)	Record.getMatrix SublistField(options)  CurrentRecord.get MatrixSublistField( options)	N/record Module  N/currentRecord Module	SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.
nlapiGetLineItemMatrixValue(type, fldnam, linenum, column)	Record.getMatrix SublistValue(options)  CurrentRecord.get MatrixSublistValue( options)	N/record Module  N/currentRecord Module	SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.
nlapiGetLineItemText(type, fldnam, linenum)	Record.getSublistText (options)  CurrentRecord. getSublistText(options)	N/record Module  N/currentRecord Module	SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.
nlapiGetLineItemValue(type, fldnam, linenum)	Record.getSublistValue (options)  CurrentRecord. getSublistValue (options)	N/record Module  N/currentRecord Module	SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.
nlapiGetLineItemValues(type, fldname, linenum)	Record.getSublistValue (options)	N/record Module	Method returns an array for multi-select fields.

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
	CurrentRecord.getSublistValue(options)	N/currentRecord Module	SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.
nlapiGetLocation()	User.location	N/runtime Module	Note that <b>location</b> is a property.
nlapiGetLogin()	auth.changeEmail(options) auth.changePassword(options)	N/auth Module	For a script sample, see <a href="#">N/auth Module Script Sample</a> .
nlapiGetMatrixCount(type, fldnam)	Record.getMatrixHeaderCount(options) CurrentRecord.getMatrixHeaderCount(options)	N/record Module N/currentRecord Module	-
nlapiGetMatrixField(type, fldnam, column)	Record.getMatrixHeaderField(options) CurrentRecord.getMatrixHeaderField(options)	N/record Module N/currentRecord Module	-
nlapiGetMatrixValue(type, fldnam, column)	Record.getMatrixHeaderValue(options) CurrentRecord.getMatrixHeaderValue(options)	N/record Module N/currentRecord Module	-
nlapiGetNewRecord()	See Notes	See Notes	To mimic this functionality in SuiteScript 2.0, use the following code in a <code>beforeLoad(scriptContext)</code> , <code>beforeSubmit(scriptContext)</code> , or <code>afterSubmit(scriptContext)</code> user event script.  <pre>function afterSubmit(context) {     var newRec = context.newRecord; }</pre> <p>For additional information and a full script sample, see the help topic <a href="#">SuiteScript 2.0 User Event Script Type</a></p>
nlapiGetOldRecord()	See Notes	See Notes	To mimic this functionality in SuiteScript 2.0, use the following code in a <code>beforeSubmit(scriptContext)</code> or <code>afterSubmit(scriptContext)</code> user event script.  <pre>function afterSubmit(context) {     var oldRec = context.oldRecord; }</pre> <p>For additional information and a full script sample, see the help topic <a href="#">SuiteScript 2.0 User Event Script Type</a></p>
nlapiGetRecordId()	Record.id CurrentRecord.id	N/record Module N/currentRecord Module	-
nlapiGetRecordType()	Record.type CurrentRecord.type	N/record Module N/currentRecord Module	To get the current record type in a client script, use <code>CurrentRecord.type</code> :  <pre>function saveRec(context) {     var rec = context.currentRecord;     var recType = rec.type; }</pre> <p>To get the current record type in a server-side script, use <code>Record.type</code> in a <code>beforeLoad(scriptContext)</code>, <code>beforeSubmit(scriptContext)</code>, or <code>afterSubmit(scriptContext)</code> user event script:</p>

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
			<pre>function beforeSubmit(context) {     var newRec = context.newRecord;     var recType = newRec.type; }</pre>
nlapiGetRole()	User.role	N/runtime Module	-
nlapiGetSubsidiary()	User.subsidiary	N/runtime Module	-
nlapi GetUser()	runtime.getCurrentUser()	N/runtime Module	-
nlapiInitiateWorkflow(recordtype, id, workflowid, initialvalues)	workflow.initiate(options)	N/workflow Module	For a script sample, see <a href="#">N/workflow Module Script Sample</a> .
nlapiInitiateWorkflowAsync(recordType, id, workflowId, initialValues)	task.WorkflowTrigger Task	N/task Module	-
nlapiInsertLineItem(type, line)	Record.insertLine(options) CurrentRecord.insertLine(options)	N/record Module N/currentRecord Module	-
nlapiInsertLineItemOption(type, fldnam, value, text, selected)	Field.insertSelectOption(options)	N/currentRecord Module	-
nlapiInsertSelectOption(fldnam, value, text, selected)	Field.insertSelectOption(options)	N/currentRecord Module	-
nlapiIsLineItemChanged(type)	Sublist.isChanged	N/record Module	Note that <code>isChanged</code> is a property  <pre>record.getSublist("addressbook").isChanged</pre>
nlapiLoadConfiguration(type)	config.load(options)	N/config Module	For a script sample, see <a href="#">N/config Module Script Sample</a> .
nlapiLoadFile(id)	file.load(options)	N/file Module	For a script sample, see <a href="#">N/file Module Script Samples</a> .
nlapiLoadRecord(type, id, initializeValues)	record.load(options)	N/record Module	-
nlapiLoadSearch(type, id)	search.load(options)	N/search Module	For a script sample, see <a href="#">N/search Module Script Samples</a> .
nlapiLogExecution(type, title, details)	log.audit(options) log.debug(options) log.emergency(options) log.error(options)	N/log Module	For a script sample, see <a href="#">N/log Module Script Sample</a> .
nlapiLookupField(type, id, fields, text)	search.lookupFields(options)	N/search Module	-
nlapiOutboundSSO(id)	sso.generateSuiteSignOnToken(options)		For a script sample, see <a href="#">N/sso Module Script Sample</a> .
nlapiPrintRecord(type, id, mode, properties)	render.bom(options) render.packingSlip(options) render.pickingTicket(options) render.statement(options) render.transaction(options)	N/render Module	For a script sample, see <a href="#">N/render Module Script Sample</a> .

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlapiRefreshLineItems(type)	-	-	This API does not have a SuiteScript 2.0 equivalent.
nlapiRefreshPortlet()	portlet.refresh	N/portlet Module	For a script sample, see <a href="#">N/portlet Module Script Sample</a>
nlapiRemoveCurrentLineItemSubrecord(sublist, fldname)	Record.removeCurrentSublistSubrecord(options) CurrentRecord.removeCurrentSublistSubrecord(options)	N/record Module N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .
nlapiRemoveLineItem(type, line)	Record.removeLine(options) CurrentRecord.removeLine(options)	N/record Module N/currentRecord Module	-
nlapiRemoveLineItemOption(type, fldnam, value)	Field.removeSelectOption(options)	N/currentRecord Module	-
nlapiRemoveSelectOption(fldnam, value)	Field.removeSelectOption(options)	N/currentRecord Module	-
nlapiRemoveSubrecord(fldname)	Record.removeSubrecord(options) CurrentRecord.removeSubrecord(options)	N/record Module N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .
nlapiRequestURL(url, postdata, headers, callback, httpMethod)	http.delete(options) http.get(options) http.post(options) http.put(options) http.request(options)	N/http Module	-
nlapiRequestURLWithCredentials(credentials, url, postdata, headers, httpsMethod)	https.request(options)	N/https Module	Server-side scripts only
nlapiResizePortlet()	portlet.resize	N/portlet Module	For a script sample, see <a href="#">N/portlet Module Script Sample</a>
nlapiResolveURL(type, identifier, id, displayMode)	url.resolveRecord(options) url.resolveScript(options) url.resolveTaskLink(options)	N/url Module	For a script sample, see <a href="#">N/url Module Script Samples</a> .
nlapiScheduleScript(scriptId, deployId, params)	task.create(options)	N/task Module	<pre>var scheduleScriptTaskObj = task.create({   taskType:     task.TaskType.SCHEDULED_SCRIPT,   //Other Params });</pre>
nlapiSearchDuplicate(type, fields, id)	search.duplicates(options)	N/search Module	-
nlapiSearchGlobal(keywords)	search.global(options)	N/search Module	-
nlapiSearchRecord(type, id, filters, columns)	search.create(options) search.load(options)	N/search Module	For a script sample, see <a href="#">N/search Module Script Samples</a> .
nlapiSelectLineItem(type, linenum)	Record.selectLine(options) CurrentRecord.selectLine(options)	N/record Module N/currentRecord Module	SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlapiSelectNewLineItem(type)	Record.selectNewLine(options) CurrentRecord.selectNewLine(options)	N/record Module N/currentRecord Module	-
nlapiSelectNode(node, xpath)	XPath.select(options)	N/xml Module	-
nlapiSelectNodes(node, xpath)	XPath.select(options)	N/xml Module	-
nlapiSelectValue(node, xpath)	See Notes	N/xml Module	To mimic this functionality in SuiteScript 2.0, select a node with <a href="#">XPath.select(options)</a> and then inspect the <a href="#">Node.textContent</a> property.
nlapiSelectValues(node, path)	See Notes	N/xml Module	To mimic this functionality in SuiteScript 2.0, select an array of nodes with <a href="#">XPath.select(options)</a> and then loop through each node's <a href="#">Node.textContent</a> property.
nlapiSendCampaignEmail(campaigneventid, recipientid)	email.sendCampaignEvent(options)	N/email Module	-
nlapiSendEmail(author, recipient, subject, body, cc, bcc, records, attachments, notifySenderOnBounce, internalOnly, replyTo)	email.send(options) email.sendBulk(options)	N/email Module	For a script sample, see <a href="#">N/email Module Script Sample</a> .
nlapiSendFax(author, recipient, subject, body, records, attachments)	N/A	-	This API does not have a SuiteScript 2.0 equivalent.
nlapiSetCurrentLineItemDate TimeValue(type, fieldId, dateTIme, timeZone)	See Notes	N/format Module	Use the <a href="#">N/format</a> module to mimic this functionality in SuiteScript 2.0.
nlapiSetCurrentLineItemMatrix Value(type, fldnam, column, value, firefieldchanged, synchronous)	Record.setCurrentMatrixSublistValue(options) CurrentRecord.set CurrentMatrixSublistValue(options)	N/record Module N/currentRecord Module	-
nlapiSetCurrentLineItemText(type, fldnam, text, firefieldchanged, synchronous)	Record.setCurrentSublistText(options) CurrentRecord.set CurrentSublistText(options)	N/record Module N/currentRecord Module	-
nlapiSetCurrentLineItemValue(type, fldnam, value, firefieldchanged, synchronous)	Record.setCurrentSublistValue(options) CurrentRecord.set CurrentSublistValue(options)	N/record Module N/currentRecord Module	-
nlapiSetCurrentLineItemValues(type, fldnam, values, firefieldchanged, synchronous)	Record.setCurrentSublistValue(options) CurrentRecord.set CurrentSublistValue(options)	N/record Module N/currentRecord Module	-
nlapiSetDateTimeValue(fieldId, dateTIme, timeZone)	See Notes	N/format Module	Use the <a href="#">N/format</a> module to mimic this functionality in SuiteScript 2.0.
nlapiSetFieldText(fldname, txt, firefieldchanged, synchronous)	Record.setText(options) CurrentRecord.setText(options)	N/record Module N/currentRecord Module	-
nlapiSetFieldTexts (fldname, txts, firefieldchanged, synchronous)	Record.setText(options) CurrentRecord.setText(options)	N/record Module N/currentRecord Module	-

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlapiSetFieldValue(fldnam, value, firefieldchanged, synchronous)	Record.setValue (options)  CurrentRecord. setValue(options)	N/record Module  N/currentRecord Module	-
nlapiSetFieldValues (fldnam, value, firefieldchanged, synchronous)	Record.setValue (options)  CurrentRecord. setValue(options)	N/record Module  N/currentRecord Module	-
nlapiSetLineItemDateTimeValue(type, fieldId, lineNum, dateTime, timeZone)	See Notes	N/format Module	Use the <b>N/format</b> module to mimic this functionality in SuiteScript 2.0.
nlapiSetLineItemValue(type, fldnam, linenum, value)	Record.setSublistValue (options)	N/record Module	SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.
nlapiSetMatrixValue(type, fldnam, column, value, firefieldchanged, synchronous)	Record.setMatrix HeaderValue(options)  CurrentRecord.set MatrixHeaderValue( options)	N/record Module  N/currentRecord Module	-
nlapiSetRecoveryPoint()	See Notes	See Notes	The <a href="#">SuiteScript 2.0 Map/Reduce Script Type</a> automatically incorporates yielding.
nlapiSetRedirectURL(type, identifier, id, editmode, parameters)	redirect.redirect (options)  redirect.toRecord (options)  redirect.toSuitelet (options)  redirect.toTaskLink (options)	N/redirect Module	For a script sample, see <a href="#">N/redirect Module Script Samples</a> .
nlapiStringToDate(str, format)	format.parse(options)	N/format Module	For a script sample, see <a href="#">N/format Module Script Samples</a> .
nlapiStringToXML(text)	Parser.fromString (options)	N/xml Module	-
nlapiSubmitConfiguration(name)	Record.save(options)	N/record Module	-
nlapiSubmitCSVImport (nlobjCSVImport)		N/task Module	-
nlapiSubmitRecord(record, doSourcing, ignoreMandatoryFields)	Record.save(options)	N/record Module	-
nlapiSubmitField(type, id, fields, values, doSourcing)	record.submitFields (options)	N/record Module	-
nlapiSubmitFile(file)	File.save()	N/file Module	For a script sample, see <a href="#">N/file Module Script Samples</a> .
nlapiTransformRecord(type, id, transformType, transformValues)	record.transform (options)	N/record Module	-
nlapiTriggerWorkflow(recordtype, id, workflowid, actionid, stateid)	workflow.trigger (options)	N/workflow Module	-
nlapiValidateXML(xmlDocument, schemaDocument, schemaFolderId)	xml.validate(options)	N/xml Module	-
nlapiViewCurrentLineItemSubrecord(sublist, fldname)	CurrentRecord. getCurrentSublist Subrecord(options)  Record.getCurrent SublistSubrecord( options)	N/record Module  N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlapiViewLineItemSubrecord(sublist, fldname, linenum)	Record.getSublist Subrecord(options)	N/record Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .  SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.
nlapiViewSubrecord(fldname)	Record.getSubrecord (options)  CurrentRecord. getSubrecord(options)	N/record Module  N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .
nlapiVoidTransaction(transactionType, recordId)	transaction. void(options)	N/transaction Module	For a script sample, see <a href="#">N/transaction Module Script Samples</a>
nlapiXMLToPDF(xmlstring)	render.xmlToPdf (options)  TemplateRenderer. renderAsPdf()	N/render Module	Note that <code>TemplateRenderer.renderAsPdf()</code> is equivalent to <code>nlapiXMLToPDF(nlobjEmailMerger.renderToString())</code> .  For a script sample, see <a href="#">N/render Module Script Sample</a> .
nlapiXMLToString(xml)	Parser.toString (options)	N/xml Module	-
nlapiYieldScript()	See Notes	See Notes	Note that the <a href="#">SuiteScript 2.0 Map/Reduce Script Type</a> automatically incorporates yielding.

## 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.

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

**i 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](#)
- [nlobjAssistantStep](#)
- [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](#)

## nlobjAssistant

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
<a href="#">nlobjAssistant</a>	<a href="#">serverWidget.Assistant</a>	N/ui/serverWidget Module	-
<a href="#">nlobjAssistant.addField(name, type, label, source, group)</a>	<a href="#">Assistant.addField(options)</a>	N/ui/serverWidget Module	-
<a href="#">nlobjAssistant.addFieldGroup(name, label)</a>	<a href="#">Assistant.addFieldGroup(options)</a>	N/ui/serverWidget Module	-
<a href="#">nlobjAssistant.addStep(name, label)</a>	<a href="#">Assistant.addStep(options)</a>	N/ui/serverWidget Module	-
<a href="#">nlobjAssistant.addSubList(name, type, label)</a>	<a href="#">Assistant.addSublist(options)</a>	N/ui/serverWidget Module	-
<a href="#">nlobjAssistant.getAllFields()</a>	<a href="#">Assistant.getFieldIds()</a>	N/ui/serverWidget Module	-
<a href="#">nlobjAssistant.getAllFieldGroups()</a>	<a href="#">Assistant.getFieldGroupIds()</a>	N/ui/serverWidget Module	-
<a href="#">nlobjAssistant.getAllSteps()</a>	<a href="#">Assistant.getSteps()</a>	N/ui/serverWidget Module	-
<a href="#">nlobjAssistant.getAllSubLists()</a>	<a href="#">Assistant.getSublistIds()</a>	N/ui/serverWidget Module	-
<a href="#">nlobjAssistant.getCurrentStep()</a>	<a href="#">Assistant.currentStep</a>	N/ui/serverWidget Module	Note that <b>currentStep</b> is a property.

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjAssistant.getField(name)	Assistant.getField(options)	N/ui/serverWidget Module	-
nlobjAssistant.getFieldGroup(name)	Assistant.getFieldGroup(options)	N/ui/serverWidget Module	-
nlobjAssistant.getLastAction()	Assistant.getLastAction()	N/ui/serverWidget Module	-
nlobjAssistant.getLastStep()	Assistant.getLastStep()	N/ui/serverWidget Module	-
nlobjAssistant.getNextStep()	Assistant.getNextStep()	N/ui/serverWidget Module	-
nlobjAssistant.getStep(name)	Assistant.getStep(options)	N/ui/serverWidget Module	-
nlobjAssistant.getSubList(name)	Assistant.getSublist(options)	N/ui/serverWidget Module	-
nlobjAssistant.hasError()	Assistant.hasErrorHtml()	N/ui/serverWidget Module	-
nlobjAssistant.isFinished()	Assistant.isFinished()	N/ui/serverWidget Module	-
nlobjAssistant.sendRedirect(response)	Assistant.sendRedirect(options)	N/ui/serverWidget Module	-
nlobjAssistant.setCurrentStep(step)	Assistant.currentStep	N/ui/serverWidget Module	Note that <b>currentStep</b> is a property.
nlobjAssistant.setError(html)	Assistant.errorHtml	N/ui/serverWidget Module	Note that <b>errorHtml</b> is a property.
nlobjAssistant.setFieldValues(values)	Assistant.updateDefaultValues(values)	N/ui/serverWidget Module	-
nlobjAssistant.setFinished(html)	Assistant.finishedHtml	N/ui/serverWidget Module	Note that <b>finishedHtml</b> is a property.
nlobjAssistant.setNumbered(hasStepNumber)	Assistant.hideStepNumber	N/ui/serverWidget Module	Note that <b>hideStepNumber</b> is a property.
nlobjAssistant.setOrdered(ordered)	Assistant.isNotOrdered	N/ui/serverWidget Module	Note that <b>isNotOrdered</b> is a property.
nlobjAssistant.setScript(script)	Assistant.clientScriptFileId Assistant.clientScriptModulePath	N/ui/serverWidget Module	Note that <b>clientScriptFileId</b> and <b>clientScriptModulePath</b> are properties.  Use one of these SuiteScript 2.0 properties to attach an ad hoc client script to an assistant.
nlobjAssistant.setShortcut(show)	Assistant.hideAddToShortcutsLink	N/ui/serverWidget Module	Note that <b>hideAddToShortcutsLink</b> is a property.
nlobjAssistant.setSplash(title, text1, text2)	Assistant.setSplash(options)	N/ui/serverWidget Module	-
nlobjAssistant.setTitle(title)	Assistant.title	N/ui/serverWidget Module	Note that <b>title</b> is a property.

## nlobjAssistantStep

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjAssistantStep	serverWidget.AssistantStep	N/ui/serverWidget Module	-
nlobjAssistantStep.getAllFields()	AssistantStep.getFieldIds()	N/ui/serverWidget Module	-
nlobjAssistantStep.getAllLineItemFields( (group))	AssistantStep.getSublistFieldIds( (options))	N/ui/serverWidget Module	-
nlobjAssistantStep.getAllLineItems()	AssistantStep. getSubmittedSublistIds()	N/ui/serverWidget Module	-
nlobjAssistantStep.getFieldValue(name)	AssistantStep.getValue(options)	N/ui/serverWidget Module	-
nlobjAssistantStep.getFieldValues(name)	AssistantStep.getValue(options)	N/ui/serverWidget Module	-
nlobjAssistantStep.getLineItemCount( (group))	AssistantStep. getLineCount(options)	N/ui/serverWidget Module	-
nlobjAssistantStep.getLineItemValue(gro name, line)	AssistantStep.getSublistValue( (options))	N/ui/serverWidget Module	-
nlobjAssistantStep.getStepNumber()	AssistantStep.stepNumber	N/ui/serverWidget Module	Note that <code>stepNumber</code> is a property.
nlobjAssistantStep.setHelpText(help)	AssistantStep.helpText	N/ui/serverWidget Module	Note that <code>helpText</code> is a property.
nlobjAssistantStep.setLabel(label)	AssistantStep.label	N/ui/serverWidget Module	Note that <code>label</code> is a property.

## nlobjButton

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjButton	serverWidget.Button	N/ui/serverWidget Module	-
nlobjButton.setDisabled( (disabled))	Button.isDisabled	N/ui/serverWidget Module	Note that <code>isDisabled</code> is a property.
nlobjButton.setLabel(label)	Button.label	N/ui/serverWidget Module	Note that <code>label</code> is a property.
nlobjButton.setVisible(visible)	Button.isHidden	N/ui/serverWidget Module	Note that <code>isHidden</code> is a property.

## nlobjColumn

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjColumn	serverWidget.ListColumn	N/ui/serverWidget Module	-

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjColumn.addParamToURL(param, value, dynamic)	ListColumn.addParamToURL(options)	N/ui/serverWidget Module	-
nlobjColumn.setLabel(label)	ListColumn.label	N/ui/serverWidget Module	Note that <code>label</code> is a property.
nlobjColumn.setURL(url, dynamic)	ListColumn.setURL(options)	N/ui/serverWidget Module	-

## nlobjConfiguration

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjConfiguration	record.Record	N/record Module	Use the <a href="#">N/config Module</a> method, <code>config.load(options)</code> , to return a <code>record.Record</code> object. Then use the <code>record.Record</code> object members to access the specified configuration page.  For a script sample, see <a href="#">N/config Module Script Sample</a> .
nlobjConfiguration.getAllFields()	Record.getFields()	N/record Module	-
nlobjConfiguration.getField(fldname)	Record.getField(options)	N/record Module	-
nlobjConfiguration.getFieldText(name)	Record.getText(options)	N/record Module	-
nlobjConfiguration.getFieldTexts(name)	Record.getText(options)	N/record Module	-
nlobjConfiguration.getFieldValue(name)	Record.getValue(options)	N/record Module	-
nlobjConfiguration.getFieldValues(name)	Record.getValue(options)	N/record Module	-
nlobjConfiguration.getType()	Record.type	N/record Module	Note that <code>type</code> is a property.
nlobjConfiguration.setTextText(name)	Record.setText(options)	N/record Module	-
nlobjConfiguration.setTextTexts(name)	Record.setText(options)	N/record Module	-
nlobjConfiguration.setValue(name)	Record.setValue(options)	N/record Module	-
nlobjConfiguration.setValueValues(name)	Record.setValue(options)	N/record Module	-

## nlobjContext

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjContext	runtime.Script runtime.Session runtime.User	N/runtime Module	-
nlobjContext.getCompanyId()	runtime.accountId	N/runtime Module	Note that <code>accountId</code> is a property.

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjContext.getDepartment()	User.department	N/runtime Module	Note that <b>department</b> is a property.
nlobjContext.getDeploymentId()	Script.deploymentId	N/runtime Module	Note that <b>deploymentId</b> is a property.
nlobjContext.getEmail()	User.email	N/runtime Module	Note that <b>email</b> is a property.
nlobjContext.getEnvironment()	runtime.envType	N/runtime Module	Note that <b>envType</b> is a property.
nlobjContext.getExecutionContext()	runtime.executionContext	N/runtime Module	Note that <b>executionContext</b> is a property.
nlobjContext.getFeature(name)	runtime.isFeatureInEffect(options)	N/runtime Module	-
nlobjContext.getLocation()	User.location	N/runtime Module	Note that <b>location</b> is a property.
nlobjContext.getLogLevel()	Script.logLevel	N/runtime Module	Note that <b>logLevel</b> is a property.
nlobjContext.getName()	User.name	N/runtime Module	Note that <b>name</b> is a property.
nlobjContext.getPercentComplete()	Script.percentComplete	N/runtime Module	Note that <b>percentComplete</b> is a property.  For a script sample, see <a href="#">N/runtime Module Script Samples</a> .
nlobjContext.getPermission(name)	User.getPermission(options)	N/runtime Module	-
nlobjContext.getPreference(name)	User.getPreference(options)	N/runtime Module	-
nlobjContext.getQueueCount()	runtime.queueCount	N/runtime Module	Note that <b>queueCount</b> is a property.
nlobjContext.getRemainingUsage()	Script.getRemainingUsage()	N/runtime Module	-
nlobjContext.getRole()	User.role	N/runtime Module	Note that <b>role</b> is a property.
nlobjContext.getRoleCenter()	User.roleCenter	N/runtime Module	Note that <b>roleCenter</b> is a property.
nlobjContext.getRoleId()	User.roleId	N/runtime Module	Note that <b>roleId</b> is a property.
nlobjContext.getScriptId()	Script.id	N/runtime Module	Note that <b>id</b> is a property.
nlobjContext.getSessionObject(name)	Session.get(options)	N/runtime Module	-
nlobjContext.getSetting(type, name)	Script.getParameter(options) Session.get(options) runtime.isFeatureInEffect(options) User.getPermission(options)	N/runtime Module	The method <b>Script.getParameter(options)</b> is equivalent to <b>nlobjContext.getSetting('SCRIPT', name)</b> .  The method <b>Session.get(options)</b> is equivalent to <b>nlobjContext.getSetting('SESSION', name)</b> .  The method <b>runtime.isFeatureInEffect(options)</b> is equivalent to <b>nlobjContext.getSetting('FEATURE', name)</b> .  The method <b>User.getPermission(options)</b> is equivalent to <b>nlobjContext.getSetting('USER', name)</b> .

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
			getSetting('PERMISSION', name).
nlobjContext.getSubsidiary()	User.subsidiary	N/runtime Module	Note that <b>subsidiary</b> is a property.
nlobjContext.getUser()	User.id	N/runtime Module	Note that <b>id</b> is a property.
nlobjContext.getVersion()	runtime.version	N/runtime Module	Note that <b>version</b> is a property.
nlobjContext.setPercentCompleteScript.percentComplete (pct)	Session.set(options)	N/runtime Module	Note that <b>percentComplete</b> is a property.
nlobjContext.setSessionObject (name, value)	Session.set(options)	N/runtime Module	-
nlobjContext.setSetting(type, name, value)	Session.set(options)	N/runtime Module	-

## nlobjCredentialBuilder

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjCredentialBuilder(string, domainString)	https.SecureString	N/https Module	-
nlobjCredentialBuilder.append (nlobjCredentialBuilder)	SecureString.appendSecureString (options) SecureString.appendString(options)	N/https Module	-
nlobjCredentialBuilder.base64()	SecureString.convertEncoding (options)	N/https Module	-
nlobjCredentialBuilder.md5()	SecureString.hash(options)	N/https Module	-
nlobjCredentialBuilder.replace(string1, string2)	-	N/https Module	There is not an equivalent of this API in SuiteScript 2.0.
nlobjCredentialBuilder.sha1()	SecureString.hash(options)	N/https Module	-
nlobjCredentialBuilder.sha256()	SecureString.hash(options)	N/https Module	-
nlobjCredentialBuilder.utf8()	SecureString.convertEncoding (options)	N/https Module	-

## nlobjCSVImport

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjCSVImport	task.CsvImportTask	N/task Module	<p>Returned by <code>task.create(options)</code>.</p> <pre>var csvImpTaskObj = task.create({     taskType:         task.TaskType.CSV_IMPORT,     //Other Params });</pre>

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjCSVImport.setLinkedFile(sublist, file)	CsvImportTask.linkedFiles	N/task Module	Note that <code>linkedFiles</code> is a property.
nlobjCSVImport.setMapping (savedImport)	CsvImportTask.mappingId	N/task Module	Note that <code>mappingId</code> is a property.
nlobjCSVImport.setOption(option, value)	CsvImportTask.name	N/task Module	Note that <code>name</code> is a property.
nlobjCSVImport.setPrimaryFile(file)	CsvImportTask.importFile	N/task Module	Note that <code>importFile</code> is a property.
nlobjCSVImport.setQueue(string)	CsvImportTask.queueId	N/task Module	Note that <code>queueId</code> is a property.

## nlobjDuplicateJobRequest

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjDuplicateJobRequest	task.EntityDeduplicationTask	N/task Module	Returned by <code>task.create(options)</code> .  <pre>var dedupTaskObj = task.create({     taskType: task.TaskType.ENTITY_DEDUPLICATION,     //Other Params });</pre>
nlobjDuplicateJobRequest.setEntityType(entityType)	EntityDeduplicationTask.entityType	N/task Module	Note that <code>entityType</code> is a property.
nlobjDuplicateJobRequest.setMasterId(masterID)	EntityDeduplicationTask.masterRecordId	N/task Module	-
nlobjDuplicateJobRequest.setMasterSelectionMode(mode)	EntityDeduplicationTask.masterSelectionMode	N/task Module	Note that <code>masterSelectionMode</code> is a property.
nlobjDuplicateJobRequest.setOperation(operation)	EntityDeduplicationTask.dedupeMode	N/task Module	Note that <code>dedupeMode</code> is a property.
nlobjDuplicateJobRequest.setRecords(dupeRecords)	EntityDeduplicationTask.recordIds	N/task Module	Note that <code>recordIds</code> is a property.

## nlobjEmailMerger

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjEmailMerger	render.EmailMergeResult	N/render Module	-
nlobjEmailMerger.merge()	See Notes	N/render Module	In SuiteScript 2.0, this is automatically called in <code>render.mergeEmail(options)</code> .
nlobjEmailMerger.setCustomRecord (recordType, recordId)	See Notes	N/render Module	In SuiteScript 2.0, this value is set with a <code>render.mergeEmail(options)</code> parameter.

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjEmailMerger.setEntity(entityType, entityId)	See Notes	N/render Module	In SuiteScript 2.0, this value is set with a <a href="#">render.mergeEmail(options)</a> parameter.
nlobjEmailMerger.setRecipient(recipientType, recipientId)	See Notes	N/render Module	In SuiteScript 2.0, this value is set with a <a href="#">render.mergeEmail(options)</a> parameter.
nlobjEmailMerger.setSupportCase(caseId)	See Notes	N/render Module	In SuiteScript 2.0, this value is set with a <a href="#">render.mergeEmail(options)</a> parameter.
nlobjEmailMerger.setTransaction(transactionId)	See Notes	N/render Module	In SuiteScript 2.0, this value is set with a <a href="#">render.mergeEmail(options)</a> parameter.

## nlobjError

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjError	error.SuiteScriptError error.UserEventError	N/error Module	-
nlobjError.getCode()	SuiteScriptError.name or UserEventError.name	N/error Module	Note that <a href="#">SuiteScriptError.name</a> and <a href="#">UserEventError.name</a> are properties.
nlobjError.getDetails()	SuiteScriptError.message or UserEventError.message	N/error Module	Note that <a href="#">SuiteScriptError.message</a> and <a href="#">UserEventError.message</a> are properties.
nlobjError.getId()	SuiteScriptError.id or UserEventError.id	N/error Module	Note that <a href="#">SuiteScriptError.id</a> and <a href="#">UserEventError.id</a> are properties.
nlobjError.getInternalId()	UserEventError.recordId	N/error Module	Note that <a href="#">UserEventError.recordId</a> is a property.
nlobjError.getStackTrace()	SuiteScriptError.stack or UserEventError.stack	N/error Module	Note that <a href="#">SuiteScriptError.stack</a> and <a href="#">UserEventError.stack</a> are properties.
nlobjError.getUserEvent()	UserEventError.eventType	N/error Module	Note that <a href="#">UserEventError.eventType</a> is a property.

## nlobjField

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjField	serverWidget.Field record.Field	N/ui/serverWidget Module N/record Module	Use the <b>N/ui/serverWidget</b> module to create and modify form fields in a Suitelet.  Use the <b>N/record</b> module to access field metadata in client and server-side scripts.

## nlobjFieldGroup

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjFieldGroup	serverWidget.FieldGroup	N/ui/serverWidget Module	-
nlobjFieldGroup.setCollapsible( <i>collapsible</i> , <i>hidden</i> )	FieldGroup.isCollapsible	N/ui/serverWidget Module	Note that <b>isCollapsible</b> is a property.
nlobjFieldGroup.setLabel( <i>label</i> )	FieldGroup.label	N/ui/serverWidget Module	Note that <b>label</b> is a property.
nlobjFieldGroup setShowBorder( <i>show</i> )	FieldGroup. isBorderHidden	N/ui/serverWidget Module	Note that <b>isBorderHidden</b> is a property.
nlobjFieldGroup.setSingleColumn( <i>column</i> )	FieldGroup. isSingleColumn	N/ui/serverWidget Module	Note that <b>isSingleColumn</b> is a property.

## nlobjFile

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjFile	file.File	N/file Module	For a script sample, see N/file Module Script Sample.
nlobjFile.getDescription()	File.description	N/file Module	Note that <b>description</b> is a property.
nlobjFile.getFolder()	File.folder	N/file Module	Note that <b>folder</b> is a property.  For a script sample, see N/file Module Script Sample.
nlobjFile.getId()	File.id	N/file Module	Note that <b>id</b> is a property.  For a script sample, see N/file Module Script Sample.
nlobjFile.getName()	File.name	N/file Module	Note that <b>name</b> is a property.  For a script sample, see N/file Module Script Sample.
nlobjFile.getSize()	File.size	N/file Module	Note that <b>size</b> is a property.
nlobjFile.getType()	File.fileType	N/file Module	Note that <b>fileType</b> is a property.

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
			For a script sample, see N/file Module Script Sample. For a script sample, see N/file Module Script Sample.
nlobjFile.getURL()	File.url	N/file Module	Note that <b>url</b> is a property.
nlobjFile.getValue()	File.getContents()	N/file Module	-
nlobjFile.isInactive()	File.isInactive	N/file Module	Note that <b>isInactive</b> is a property.
nlobjFile.isOnline()	File.isOnline	N/file Module	Note that <b>isOnline</b> is a property. For a script sample, see N/file Module Script Sample.
nlobjFile.setDescription( <i>description</i> )	File.description	N/file Module	Note that <b>description</b> is a property.
nlobjFile.setEncoding( <i>encodingType</i> )	File.encoding	N/file Module	Note that <b>encoding</b> is a property.
nlobjFile.setFolder( <i>id</i> )	File.folder	N/file Module	Note that <b>folder</b> 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.
nlobjFile.setIsInactive( <i>inactive</i> )	File.isInactive	N/file Module	Note that <b>isInactive</b> is a property.
nlobjFile.setIsOnline( <i>online</i> )	File.isOnline	N/file Module	Note that <b>isOnline</b> is a property. For a script sample, see N/file Module Script Sample.
nlobjFile.setName( <i>name</i> )	File.name	N/file Module	Note that <b>name</b> is a property. For a script sample, see N/file Module Script Sample.

## nlobjForm

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjForm	serverWidget.Form	N/ui/serverWidget Module	-
nlobjForm.addButton( <i>name</i> , <i>label</i> , <i>script</i> )	Form.addButton( <i>options</i> )	N/ui/serverWidget Module	-
nlobjForm.addCredentialField( <i>id</i> , <i>label</i> , <i>website</i> , <i>scriptId</i> , <i>value</i> , <i>entityMatch</i> , <i>tab</i> )	Form.addCredentialField( <i>options</i> )	N/ui/serverWidget Module	-

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjForm.addField(name, type, label, sourceOrRadio, tab)	Form.addField(options)	N/ui/serverWidget Module	For a script sample, see <a href="#">N/ui/serverWidget Module Script Samples</a>
nlobjForm.addFieldGroup(name, label, tab)	Form.addFieldGroup(options)	N/ui/serverWidget Module	-
nlobjForm.addPageLink(type, title, url)	Form.addPageLink(options)	N/ui/serverWidget Module	-
nlobjForm.addResetButton(label)	Form.addResetButton(options)	N/ui/serverWidget Module	-
nlobjForm.addSubList(name, type, label, tab)	Form.addSublist(options)	N/ui/serverWidget Module	For a script sample, see <a href="#">N/ui/serverWidget Module Script Samples</a>
nlobjForm.addSubmitButton(label)	Form.addSubmitButton(options)	N/ui/serverWidget Module	For a script sample, see <a href="#">N/ui/serverWidget Module Script Samples</a>
nlobjForm.addSubTab(name, label, tab)	Form.addSubtab(options)	N/ui/serverWidget Module	-
nlobjForm.addTab(name, label)	Form.addTab(options)	N/ui/serverWidget Module	-
nlobjForm.getButton(name)	Form.getButton(options)	N/ui/serverWidget Module	-
nlobjForm.getField(name, radio)	Form.getField(options)	N/ui/serverWidget Module	-
nlobjForm.getSubList(name)	Form.getSublist(options)	N/ui/serverWidget Module	-
nlobjForm.getSubTab(name)	Form.getSubtab(options)	N/ui/serverWidget Module	-
nlobjForm.getTab(name)	Form.getTab(options)	N/ui/serverWidget Module	-
nlobjForm.getTabs()	Form.getTabs()	N/ui/serverWidget Module	-
nlobjForm.insertField(field, nextfld)	Form.insertField(options)	N/ui/serverWidget Module	-
nlobjForm.insertSubList(sublist, nextsub)	Form.insertSublist(options)	N/ui/serverWidget Module	-
nlobjForm.insertSubTab(subtab, nextsub)	Form.insertSubtab(options)	N/ui/serverWidget Module	-
nlobjForm.insertTab(tab, nexttab)	Form.insertTab(options)	N/ui/serverWidget Module	-
nlobjForm.removeButton(name)	Form.removeButton(options)	N/ui/serverWidget Module	-
nlobjForm.setFieldValues(values)	Form.updateDefaultValues(options)	N/ui/serverWidget Module	-
nlobjForm.setScript(script)	Form.clientScriptFileDialog Form.clientScriptModulePath	N/ui/serverWidget Module	Note that <code>clientScriptFileDialog</code> and <code>clientScriptModulePath</code> are properties.

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
			Use one of these SuiteScript 2.0 properties to attach an ad hoc client script to a form.
nlobjForm.setTitle(title)	Form.title	N/ui/serverWidget Module	Note that <b>title</b> is a property.

## nlobjFuture

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjFuture	task.EntityDeduplicationTaskStatus	N/task Module	-
nlobjFuture.isCancelled()	EntityDeduplicationTaskStatus.status	N/task Module	Note that <b>status</b> is a property.
nlobjFuture.isDone()	EntityDeduplicationTaskStatus.status	N/task Module	Note that <b>status</b> is a property.

## nlobjJobManager

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjJobManager	task.EntityDeduplicationTaskStatus	N/task Module	-
nlobjJobManager.createJobRequest()	task.create(options)	N/task Module	-
nlobjJobManager.getFuture()	task.checkStatus(options)	N/task Module	-
nlobjJobManager.submit (nlobjDuplicateJobRequest)	EntityDeduplicationTask.submit()	N/task Module	-

## nlobjList

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjList	serverWidget.List	N/ui/serverWidget Module	-
nlobjList.addButton(name, label, script)	List.addButton(options)	N/ui/serverWidget Module	-
nlobjList.addColumn(name, type, label, align)	List.addColumn(options)	N/ui/serverWidget Module	-
nlobjList.addEditColumn(column, showView, showHrefCol)	List.addEditColumn(options)	N/ui/serverWidget Module	-
nlobjList.addPageLink(type, title, url)	List.addPageLink(options)	N/ui/serverWidget Module	-
nlobjList.addRow(row)	List.addRow(options)	N/ui/serverWidget Module	-
nlobjList.addRows(rows)	List.addRows(options)	N/ui/serverWidget Module	-

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjList.setScript(script)	List.clientScriptFileDialog List.clientScriptModulePath	N/ui/serverWidget Module	Note that <code>clientScriptFileDialog</code> and <code>clientScriptModulePath</code> are properties.  Use one of these SuiteScript 2.0 properties to attach an ad hoc client script to a form.
nlobjList.setStyle(style)	List.style	N/ui/serverWidget Module	Note that <code>style</code> is a property.
nlobjList.setTitle(title)	List.title	N/ui/serverWidget Module	Note that <code>title</code> is a property.

## nlobjLogin

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjLogin	See Notes.	N/auth Module	See nlobjLogin members.
nlobjLogin.changeEmail (currentPassword, newEmail, justThisAccount)	auth.changeEmail(options)	N/auth Module	For a script sample, see N/auth Module Script Sample.
nlobjLogin.changePassword (currentPassword, newPassword)	auth.changePassword(options)	N/auth Module	For a script sample, see N/auth Module Script Sample

## nlobjMergeResult

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjMergeResult	render.EmailMergeResult	N/render Module	-
nlobjMergeResult.getBody()	EmailMergeResult.body	N/render Module	Note that <code>body</code> is a property.
nlobjMergeResult.getSubject()	EmailMergeResult.subject	N/render Module	Note that <code>subject</code> is a property.

## nlobjPortlet

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjPortlet	Portlet Object	See Notes	For additional information, see the help topic SuiteScript 2.0 Portlet Script Type.
nlobjPortlet.addColumn(name, type, label, just)	Portlet.addColumn(options)	-	For additional information, see the help topic SuiteScript 2.0 Portlet Script Type.

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjPortlet.addEditColumn(column, showView, showHrefCol)	Portlet.addEditColumn(options)	-	For additional information, see the help topic <a href="#">SuiteScript 2.0 Portlet Script Type</a> .
nlobjPortlet.addField(name, type, label, source)	Portlet.addField(options)	-	For additional information, see the help topic <a href="#">SuiteScript 2.0 Portlet Script Type</a> .
nlobjPortlet.addLine(text, url, indent)	Portlet.addLine(options)	-	For additional information, see the help topic <a href="#">SuiteScript 2.0 Portlet Script Type</a> .
nlobjPortlet.addRow(row)	Portlet.addRow(options)	-	For additional information, see the help topic <a href="#">SuiteScript 2.0 Portlet Script Type</a> .
nlobjPortlet.addRows(rows)	Portlet.addRows(options)	-	For additional information, see the help topic <a href="#">SuiteScript 2.0 Portlet Script Type</a> .
nlobjPortlet.setHtml(html)	Portlet.html	-	For additional information, see the help topic <a href="#">SuiteScript 2.0 Portlet Script Type</a> .
nlobjPortlet.setRefreshInterval(n)	-	-	There is no SuiteScript 2.0 direct equivalent for this method. For additional information, see the help topic <a href="#">SuiteScript 2.0 Portlet Script Type</a> .
nlobjPortlet.setScript(scriptid)	Portlet.clientScriptFileDialog Portlet.clientScriptModulePath	-	For additional information, see the help topic <a href="#">SuiteScript 2.0 Portlet Script Type</a> .
nlobjPortlet.setSubmitButton(url, label, target)	Portlet.setSubmitButton(options)	-	For additional information, see the help topic <a href="#">SuiteScript 2.0 Portlet Script Type</a> .
nlobjPortlet.setTitle(title)	Portlet.title	-	For additional information, see the help topic <a href="#">SuiteScript 2.0 Portlet Script Type</a> .

## nlobjRecord

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjRecord	record.Record currentRecord.CurrentRecord	N/record Module N/currentRecord Module	-
nlobjRecord.commitLineItem(group, ignoreRecalc)	Record.commitLine(options) CurrentRecord.commitLine(options)	N/record Module N/currentRecord Module	-
nlobjRecord.createCurrentLineItemSubrecord(sublist, fldname)	Record.getCurrentSublistSubrecord(options) CurrentRecord.getCurrentSublistSubrecord(options)	N/record Module N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .
nlobjRecord.createSubrecord(fldname)	Record.getSubrecord(options) CurrentRecord.getSubrecord(options)	N/record Module N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .
nlobjRecord.editCurrentLineItemSubrecord(sublist, fldname)	Record.getCurrentSublistSubrecord(options) CurrentRecord.getCurrentSublistSubrecord(options)	N/record Module N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .
nlobjRecord.editSubrecord(fldname)	Record.getSubrecord(options) CurrentRecord.getSubrecord(options)	N/record Module N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .
nlobjRecord.findLineItemMatrixValue(group, fldnam, column, val)	Record.findMatrixSublistLineWithValue(options) CurrentRecord.findMatrixSublistLineWithValue(options)	N/record Module N/currentRecord Module	-
nlobjRecord.findLineItemValue(group, fldnam, value)	Record.findSublistLineWithValue(options) CurrentRecord.findSublistLineWithValue(options)	N/record Module N/currentRecord Module	-
nlobjRecord.getAllFields()	Record.getFields()	N/record Module	-
nlobjRecord.getAllLineItemFields(group)	Record.getSublistFields(options)	N/record Module	-
nlobjRecord.getCurrentLineItemDateTimeValue(type, fieldId, timeZone)	See Notes	N/format Module	Use the N/format module to mimic this functionality in SuiteScript 2.0.
nlobjRecord.getCurrentLineItemMatrixValue(group, fldnam, column)	Record.getCurrentMatrixSublistValue(options) CurrentRecord.getCurrentMatrixSublistValue(options)	N/record Module N/currentRecord Module	-

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjRecord.getCurrentLineItemValue(type, fldnam)	Record.getCurrentSublistValue(options) CurrentRecord.getCurrentSublistValue(options)	N/record Module N/currentRecord Module	-
nlobjRecord.getCurrentLineItemValues(type, fldnam)	Record.getCurrentSublistValue(options) CurrentRecord.getCurrentSublistValue(options)	N/record Module N/currentRecord Module	Method returns an array for multi-select fields.
nlobjRecord.getDateTimeValue(fieldId, timeZone)	See Notes	N/format Module	Use the <b>N/format</b> module to mimic this functionality in SuiteScript 2.0.
nlobjRecord.getField(fldnam)	Record.getField(options) CurrentRecord.getField(options)	N/record Module N/currentRecord Module	-
nlobjRecord.getFieldText(name)	Record.getText(options) CurrentRecord.getText(options)	N/record Module N/currentRecord Module	-
nlobjRecord.getFieldTexts(name)	Record.getText(options) CurrentRecord.getText(options)	N/record Module N/currentRecord Module	-
nlobjRecord.getFieldValue(name)	Record.getValue(options) CurrentRecord.getValue(options)	N/record Module N/currentRecord Module	-
nlobjRecord.getFieldValues(name)	Record.getValue(options) CurrentRecord.getValue(options)	N/record Module N/currentRecord Module	-
nlobjRecord.getId()	Record.id	N/record Module	-
nlobjRecord.getLineItemCount(group)	Record.getLineCount(options)	N/record Module	-
nlobjRecord.getLineItemDateValue(type, fieldId, lineNum, timeZone)	See Notes	N/format Module	Use the <b>N/format</b> module to mimic this functionality in SuiteScript 2.0.
nlobjRecord.getLineItemField(group, fldnam, linenum)	Record.getSublistField(options) CurrentRecord.getSublistField(options)	N/record Module N/currentRecord Module	-
nlobjRecord.getLineItemMatrixField(group, fldnam, linenum, column)	Record.getMatrixSublistField(options) CurrentRecord.getMatrixSublistField(options)	N/record Module N/currentRecord Module	-
nlobjRecord.getLineItemMatrixValue(group, fldnam, lineum, column)	Record.getMatrixSublistValue(options) CurrentRecord.getMatrixSublistValue(options)	N/record Module N/currentRecord Module	-
nlobjRecord.getLineItemText(group, fldnam, linenum)	Record.getSublistText(options) CurrentRecord.getSublistText(options)	N/record Module N/currentRecord Module	-
nlobjRecord.getLineItemValue(group, name, linenum)	Record.getSublistValue(options)	N/record Module	-

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
	CurrentRecord.getSublistValue(options)	N/currentRecord Module	
nlobjRecord.getLineItemValues(type, fldnam, linenum)	Record.getSublistValue(options) CurrentRecord.getSublistValue(options)	N/record Module N/currentRecord Module	-
nlobjRecord.getMatrixCount(group, fldnam)	Record.getMatrixHeaderCount(options) CurrentRecord.getMatrixHeaderCount(options)	N/record Module N/currentRecord Module	-
nlobjRecord.getMatrixField(group, fldname, column)	Record.getMatrixHeaderField(options) CurrentRecord.getMatrixHeaderField(options)	N/record Module N/currentRecord Module	-
nlobjRecord.getMatrixValue(group, fldnam, column)	Record.getMatrixHeaderValue(options) CurrentRecord.getMatrixHeaderValue(options)	N/record Module N/currentRecord Module	-
nlobjRecord.getRecordType()	Record.type	N/record Module	-
nlobjRecord.insertLineItem(group, linenum, ignoreRecalc)	Record.insertLine(options) CurrentRecord.insertLine(options)	N/record Module N/currentRecord Module	-
nlobjRecord.removeLineItem(group, linenum, ignoreRecalc)	Record.removeLine(options) CurrentRecord.removeLine(options)	N/record Module N/currentRecord Module	-
nlobjRecord.removeCurrentLineItemSubrecord(sublist, fldname)	Record.removeCurrentSublistSubrecord(options)	N/record Module N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .
nlobjRecord.removeSubrecord(fldname)	Record.removeSubrecord(options) CurrentRecord.removeSubrecord(options)	N/record Module N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .
nlobjRecord.selectLineItem(group, linenum)	Record.selectLine(options) CurrentRecord.selectLine(options)	N/record Module N/currentRecord Module	-
nlobjRecord.selectNewLineItem(group)	Record.selectNewLine(options) CurrentRecord.selectNewLine(options)	N/record Module N/currentRecord Module	-
nlobjRecord.setCurrentLineItemDateTimeValue(type, fieldId, dateTime, timeZone)	See Notes	N/format Module	Use the <b>N/format</b> module to mimic this functionality in SuiteScript 2.0.
nlobjRecord.setCurrentLineItemMatrixValue(group, fldnam, column, value)	Record.setCurrentMatrixSublistValue(options) CurrentRecord.setCurrentMatrixSublistValue(options)	N/record Module N/currentRecord Module	-

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjRecord.setCurrentLineItemValue(group, name, value)	Record.setCurrentSublistValue(options) CurrentRecord.setCurrentSublistValue(options)	N/record Module N/currentRecord Module	-
nlobjRecord.setDateTimeValue(fieldId, dateTime, timeZone)	See Notes	N/format Module	Use the <b>N/format</b> module to mimic this functionality in SuiteScript 2.0.
nlobjRecord.setText(name, text)	Record.setText(options) CurrentRecord.setText(options)	N/record Module N/currentRecord Module	-
nlobjRecord.setTexts(name, text)	Record.setText(options) CurrentRecord.setText(options)	N/record Module N/currentRecord Module	-
nlobjRecord.setValue(name, value)	Record.setValue(options) CurrentRecord.setValue(options)	N/record Module N/currentRecord Module	-
nlobjRecord.setValue(name, value)	Record.setValue(options) CurrentRecord.setValue(options)	N/record Module N/currentRecord Module	-
nlobjRecord.setLineItemDate TimeValue(type, fieldId, lineNum, dateTime, timeZone)	See Notes	N/format Module	Use the <b>N/format</b> module to mimic this functionality in SuiteScript 2.0.
nlobjRecord.setLineItemValue (group, name, linenum, value)	Record.setSublistValue(options)	N/record Module	-
nlobjRecord.setMatrixValue(group, fldnam, column, value)	Record.setMatrixHeaderValue(options) CurrentRecord.setMatrixHeaderValue(options)	N/record Module N/currentRecord Module	-
nlobjRecord.viewCurrentLineItem Subrecord(sublist, fldname)	Record.getCurrentSublist Subrecord(options) CurrentRecord.getCurrentSublist Subrecord(options)	N/record Module N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .
nlobjRecord.viewLineItemSubrecord (sublist, fldname, linenum)	Record.getSublistSubrecord (options)	N/record Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .
nlobjRecord.viewSubrecord (fldname)	Record.getSubrecord(options) CurrentRecord.getSubrecord (options)	N/record Module N/currentRecord Module	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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a> .

## nlobjRequest

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjRequest	http.ServerRequest	N/http Module N/ https Module	-
nlobjRequest.getAllHeaders()	ServerRequest.headers	N/http Module N/ https Module	ServerRequest.headers(options) is read-only.
nlobjRequest.getAllParameters()	ServerRequest.parameters	N/http Module N/ https Module	ServerRequest.parameters(options) is read-only.
nlobjRequest.getBody()	ServerRequest.body	N/http Module N/ https Module	ServerRequest.body(options) is read-only.
nlobjRequest.getFile(id)	ServerRequest.files	N/http Module N/ https Module	ServerRequest.files(options) is read-only.
nlobjRequest.getHeader(name)	ServerRequest.headers	N/http Module N/ https Module	ServerRequest.headers(options) is read-only.
nlobjRequest.getLineItemCount(group)	ServerRequest.getLineCount(options)	N/http Module N/ https Module	-
nlobjRequest.getLineItemValue(group, name, line)	ServerRequest.getSublistValue(options)	N/http Module N/ https Module	-
nlobjRequest.getMethod()	ServerRequest.method	N/http Module N/ https Module	ServerRequest.method(options) is read-only.
nlobjRequest.getParameter(name)	ServerRequest.parameters	N/http Module N/ https Module	ServerRequest.parameters(options) is read-only.
nlobjRequest.getParameterValues(name)	ServerRequest.parameters	N/http Module N/ https Module	ServerRequest.parameters(options) is read-only.
nlobjRequest.getURL()	ServerRequest.url	N/http Module N/ https Module	ServerRequest.url is read-only.

## nlobjResponse

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjResponse	http.ServerResponse	N/http Module N/ https Module	-
nlobjResponse.addHeader(name, value)	ServerResponse.addHeader(options)	N/http Module N/ https Module	-
nlobjResponse.getAllHeaders()	ServerResponse.getHeader(options) or ServerResponse.headers	N/http Module N/ https Module	-
nlobjResponse.getBody()	ClientResponse.body	N/http Module N/ https Module	Note that ClientResponse.body is a property.
nlobjResponse.getCode()	ClientResponse.code	N/http Module N/ https Module	Note that ClientResponse.code is a property.
nlobjResponse.getError()	See Notes	N/http Module N/ https Module	There is no SuiteScript 2.0 equivalent for this method.

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjResponse.getHeader(name)	ServerResponse.getHeader(options)	N/http Module N/https Module	-
nlobjResponse.getHeaders(name)	ServerResponse.getHeader(options)	N/http Module N/https Module	If multiple values are assigned to the header name, serverResponse.getHeader(options) returns the values as an Array.
nlobjResponse.renderPDF(xmlString)	ServerResponse.renderPdf(options)	N/http Module N/https Module	-
nlobjResponse.setCDNCacheable(type)	ServerResponse.setCdnCacheable(options)	N/http Module N/https Module	-
nlobjResponse.setContentType(type, name, disposition)	See Notes	N/http Module N/https Module	There is no direct equivalent for this method in SuiteScript 2.0.
nlobjResponse.setEncoding(encodingType)	See Notes	N/http Module N/https Module	There is no direct equivalent for this method in SuiteScript 2.0.
nlobjResponse.setHeader(name, value)	ServerResponse.setHeader(options)	N/http Module N/https Module	-
nlobjResponse.sendRedirect(type, identifier, id, editmode, parameters)	ServerResponse.sendRedirect(options)	N/http Module N/https Module	-
nlobjResponse.write(output)	ServerResponse.write(options)	N/http Module N/https Module	-
nlobjResponse.writeLine(output)	ServerResponse.writeLine(options)	N/http Module N/https Module	-
nlobjResponse.writePage(pageobject)	ServerResponse.writePage(options)	N/http Module N/https Module	-

## nlobjSearch

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjSearch	search.Search	N/search Module	-
nlobjSearch.addColumn(column)	Search.columns	N/search Module	Note that <code>Search.columns</code> is a property.
nlobjSearch.addColumns(columns)	Search.columns	N/search Module	Note that <code>Search.columns</code> is a property.
nlobjSearch.addFilter(filter)	Search.filters	N/search Module	Note that <code>Search.filters</code> is a property.
nlobjSearch.addFilters(filters)	Search.filters	N/search Module	Note that <code>Search.filters</code> is a property.
nlobjSearch.deleteSearch()	search.delete(options)	N/search Module	For a script sample, see <a href="#">N/search Module Script Samples</a> .
nlobjSearch.getColumns()	Search.columns	N/search Module	Note that <code>columns</code> is a property.
nlobjSearch.getFilterExpression()	Search.filterExpression	N/search Module	Note that <code>filterExpression</code> is a property.
nlobjSearch.getFilters()	Search.filters	N/search Module	Note that <code>filters</code> is a property.

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjSearch.getId()	Search.searchId	N/search Module	Note that <b>id</b> is a property.
nlobjSearch.getIsPublic()	Search.isPublic	N/search Module	Note that <b>isPublic</b> is a property.
nlobjSearch.getscriptId()	Search.id	N/search Module	-
nlobjSearch.getSearchType()	Search.searchType	N/search Module	Note that <b>searchType</b> is a property.
nlobjSearch.runSearch()	Search.run()	N/search Module	-
nlobjSearch.saveSearch(title, scriptId)	Search.save()	N/search Module	-
nlobjSearch.setColumns(columns)	Search.columns	N/search Module	Note that <b>columns</b> is a property.
nlobjSearch.setFilterExpression(filterExpression)	Search.filterExpression	N/search Module	Note that <b>filterExpression</b> is a property.
nlobjSearch.setFilters(filters)	Search.filters	N/search Module	Note that <b>filters</b> is a property.
nlobjSearch.setIsPublic(type)	Search.isPublic	N/search Module	Note that <b>isPublic</b> is a property.
nlobjSearch.setRedirectURLToSearch()	redirect.toSavedSearch(options) redirect.toSearch(options)	N/redirect Module	-
nlobjSearch.setRedirectURLToSearchResults()	redirect.toSearchResult(options) redirect.toSavedSearchResult(options)	N/redirect Module	-

## nlobjSearchColumn

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjSearchColumn	search.Column	N/search Module	-

## nlobjSearchFilter

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjSearchFilter	search.Filter	N/search Module	-

## nlobjSearchResult

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjSearchResult	search.Result	N/search Module	-

## nlobjSearchResultSet

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjSearchResultSet	search.ResultSet	N/search Module	-

## nlobjSelectOption

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjSelectOption	See Notes	N/record Module	See mapping for <b>nlobjSelectOption</b> methods.
nlobjSelectOption.getId()	N/record: <a href="#">Field.getSelectOptions(options)</a> N/currentRecord: <a href="#">Field.getSelectOptions(options)</a>	N/record Module N/currentRecord Module	-
nlobjSelectOption.getText()	N/record: <a href="#">Field.getSelectOptions(options)</a> N/currentRecord: <a href="#">Field.getSelectOptions(options)</a>	N/record Module N/currentRecord Module	-

## nlobjSublist

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjSubList	serverWidget.Sublist	N/ui/serverWidget Module	-
nlobjSublist.addButton(name, label, script)	Sublist.addButton(options)	N/ui/serverWidget Module	-
nlobjSublist.addField(name, type, label, source)	Sublist.addField(options)	N/ui/serverWidget Module	-
nlobjSublist.addMarkAllButtons()	Sublist.addMarkAllButtons()	N/ui/serverWidget Module	-
nlobjSublist.addRefreshButton()	Sublist.addRefreshButton()	N/ui/serverWidget Module	-
nlobjSublist.getLineItemCount()	Sublist.lineCount	N/ui/serverWidget Module	Note that <b>lineCount</b> is a property
nlobjSublist.getLineItemValue (group, fldnam, linenum)	Sublist.getSublistValue(options)	N/ui/serverWidget Module	-
nlobjSublist.setAmountField(field)	Sublist.updateTotallingFieldId (options)	N/ui/serverWidget Module	-
nlobjSublist.setDisplayType(type)	Sublist.displayType	N/ui/serverWidget Module	Note that <b>displayType</b> is a property
nlobjSublist.setHelpText(help)	Sublist.helpText	N/ui/serverWidget Module	Note that <b>helpText</b> is a property
nlobjSublist.setLabel(label)	Sublist.label	N/ui/serverWidget Module	Note that <b>label</b> is a property
nlobjSublist.setLineItemValue (name, linenum,value)	Sublist.setSublistValue(options)	N/ui/serverWidget Module	-
nlobjSublist.setLineItemValues (values)	See Notes	N/ui/serverWidget Module	There is not a SuiteScript 2.0 direct equivalent for this method.

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjSublist.setUniqueField(name)	Sublist.updateUniqueId (options)	N/ui/serverWidget Module	Note that <code>uniqueFieldId</code> is a property

## nlobjSubrecord

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjSubrecord	See Notes	N/record Module	<p>SuiteScript 2.0 subrecords are returned as <code>record.Record</code> objects.</p> <p>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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a>.</p>
nlobjSubrecord.cancel()	See Notes	N/record Module	<p>SuiteScript 2.0 subrecords are returned as <code>record.Record</code> objects.</p> <p>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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a>.</p>
nlobjSubrecord.commit()	See Notes	N/record Module	<p>This API does not have a SuiteScript 2.0 equivalent. SuiteScript 2.0 subrecords are returned as <code>record.Record</code> objects.</p> <p>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 <a href="#">SuiteScript 2.0 Scripting Subrecords</a>.</p>

## 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 <code>label</code> is a property
nlobjTab.setHelpText(help)	Tab.helpText	N/ui/serverWidget Module	Note that <code>helpText</code> 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 <a href="#">N/render Module Script Sample</a> .

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjTemplateRenderer.addRecord(var, record)	TemplateRenderer.addRecord(options)	N/render Module	For a script sample, see <a href="#">N/render Module Script Sample</a> .
nlobjTemplateRenderer.addSearchResults(searchResult)	TemplateRenderer.addSearchResults(options)	N/render Module	For a script sample, see <a href="#">N/render Module Script Sample</a> .
nlobjTemplateRenderer.renderToResponse()	TemplateRenderer.renderToResponse(options)	N/render Module	-
nlobjTemplateRenderer.renderToString()	TemplateRenderer.renderAsString()	N/render Module	-
nlobjTemplateRenderer.setTemplate(template)	TemplateRenderer.templateContent	N/render Module	For a script sample, see <a href="#">N/render Module Script Sample</a> .

# SuiteScript 2.0 Global Objects

**Note:** The content in this help topic pertains to SuiteScript 2.0.

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

**Note:** The content in this help topic pertains to SuiteScript 2.0.

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:

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

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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Description</b>	Function used to create entry point scripts and custom modules in SuiteScript 2.0. For more information, see the help topics <a href="#">SuiteScript 2.0 Entry Point Script Creation and Deployment</a> and <a href="#">SuiteScript 2.0 Custom Modules</a> .  Use this <code>define()</code> 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.
<b>Returns</b>	Object
<b>Global object</b>	<code>define Object</code>
<b>Since</b>	Version 2015 Release 2

### Parameters

Parameter	Type	Required / Optional	Description	Since
moduleObject	Object	Required	A callback function or a module object	Version 2015 Release 2

## 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)`

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Function used to create entry point scripts and custom modules in SuiteScript 2.0. For more information, see the help topics <a href="#">SuiteScript 2.0 Entry Point Script Creation and Deployment</a> and <a href="#">SuiteScript 2.0 Custom Modules</a> .  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.
<b>Returns</b>	Object
<b>Global object</b>	<code>define Object</code>

<b>Since</b>	Version 2015 Release 2
--------------	------------------------

## Parameters

Parameter	Type	Required / Optional	Description	Since
id	string	optional	Defines the id of the module	Version 2015 Release 2
dependencies	string []	optional	<p>Represents any module dependencies required by the callback function.</p> <p>Use the following syntax:</p> <ul style="list-style-type: none"> <li>■ Native SuiteScript 2.0 modules: [ 'N/&lt;module name&gt;' ]</li> <li>■ Custom modules: [/&lt;path to module file in File Cabinet&gt;/&lt;module name&gt;]</li> </ul> <p>For other options, see the help topic <a href="#">Module Dependency Paths</a>.</p>	Version 2015 Release 2
moduleObject	Function   Object	required	A callback function or a module object	Version 2015 Release 2

## Errors

Error Code	Message	Thrown If
MODULE_DOES_NOT_EXIST	Module does not exist: {module path/name}	<p>The NetSuite module or custom module dependency does not exist.</p> <p>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.</p>

## 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.

```
...
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.

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

define(['N/record'],
```

```

function (record)
{
    function createPhoneCall(context)
    {
        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**.

```

// 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:

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

    return lib.makeSomething(); // return value of makeSomething function in custom module
});
```

## require Function

**Note:** The content in this help topic pertains to SuiteScript 2.0.

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](#).

Type	Name	Return Type / Value Type	Description
Function	<a href="#">require([dependencies,] callback)</a>	Void	Loads a SuiteScript 2.0 entry point script or a SuiteScript 2.0 custom module.  Executes the callback function and loads the dependencies when they are required.

**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)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.  If you add a module as a dependency and the module is never used, the dependency is never loaded.
---------------------------	---

	 <b>Note:</b> This function conforms to the Asynchronous Module Definition (AMD) specification. For more information, see <a href="#">require Function</a> .
<b>Returns</b>	Void
<b>Global object</b>	<a href="#">require Function</a>
<b>Since</b>	Version 2015 Release 2

## Parameters

Parameter	Type	Required / Optional	Description	Since
dependencies	string []	Optional	<p>Represents any module dependencies required by the callback function.</p> <p>Use the following syntax:</p> <ul style="list-style-type: none"> <li>■ Native SuiteScript 2.0 modules: [ 'N/<module name="">' ]</module></li> <li>■ Custom modules: [ '/&lt;path to module file in File Cabinet&gt;/&lt;module name&gt;' ]</li> </ul> <p>See the help topic <a href="#">Module Dependency Paths</a>.</p>	Version 2015 Release 2
callback	Function	Required	Callback function to execute. Dependent modules are not loaded until they are required.	Version 2015 Release 2

## Errors

Error Code	Message	Thrown If
MODULE_DOES_NOT_EXIST	Module does not exist: {module path/name}	<p>The NetSuite module or custom module dependency does not exist.</p> <p>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.</p>

## Syntax

The following example shows progressive loading of modules in a script. For a functional example, see [require Function](#).

```
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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

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

If you set up a valid `@NAmdConfig` 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 `@NAmdConfig` tag and provide a file cabinet path to the configuration file

```
/**
 * @NAmdConfig /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:

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



**Important:** Ensure that configuration file uses JSON syntax (and not JavaScript syntax). For more information about JSON, visit <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

SuiteScript accepts the configuration values outlined at <https://github.com/amdjs/amdjs-api/blob/master/CommonConfig.md> (version fd45c71).

You can use the JS Doc 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:

Parameter	Type	Required / Optional	Sample Usage	Since
baseUrl	string	Optional	<ul style="list-style-type: none"> <li>■ To configure a shorter relative path by indicating the root folder that holds the modules in the file cabinet.</li> </ul>	Version 2015 Release 2
paths	Object	Optional	<ul style="list-style-type: none"> <li>■ To create a named alias to a path</li> <li>■ For testing purposes, pass in an object that serves as a mock-up of another module.</li> <li>■ To set up a custom name for a SuiteScript module</li> </ul>	Version 2015 Release 2
packages	Object[]	Optional	<ul style="list-style-type: none"> <li>■ To configure a special lookup suitable for traditional CommonJS packages that you want to use as a custom module.</li> </ul>	Version 2015 Release 2
map	Object	Optional	<ul style="list-style-type: none"> <li>■ To specify an alias.</li> <li>■ To handle multiple names for a module</li> <li>■ To load a set of identically-named but unique modules, such as dependency on multiple module versions.</li> </ul>	Version 2015 Release 2
config	Object	Optional	<ul style="list-style-type: none"> <li>■ To assign attributes, such as metadata.</li> </ul>	Version 2015 Release 2
shim	Object	Optional	<ul style="list-style-type: none"> <li>■ To prepare a non-AMD JS library for loading.</li> </ul>	Version 2015 Release 2

## require.config()

Configuration of a require Object is optional and for advanced usage only. If you must configure a require Object, the **@NAmConfig** 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

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.isBoolean\(obj\)](#)
- [util.isDate\(obj\)](#)
- [utilisFunction\(obj\)](#)
- [util.isNumber\(obj\)](#)
- [utilisObject\(obj\)](#)
- [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()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.   <b>Note:</b> Consider this method a replacement for the instanceof 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 instanceof, an undefined error is thrown.
<b>Returns</b>	The object type as a string
<b>Supported Script Types</b>	All script types

<b>Governance</b>	None
<b>Since</b>	Version 2015 Release 2

## Syntax

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

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

# JSON object



**Note:** The content in this help topic pertains to SuiteScript 2.0.

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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Parse a string as a JSON object and returns the object.  The text parameter must conform to the JSON standard. See <a href="http://www.ietf.org/rfc/rfc4627.txt">http://www.ietf.org/rfc/rfc4627.txt</a> .
<b>Returns</b>	Object
<b>Supported Script Types</b>	All script types
<b>Governance</b>	None
<b>Global object</b>	<a href="#">JSON object</a>
<b>Since</b>	Version 2015 Release 2

## Parameters

Parameter	Type	Required / Optional	Description	Since
text	string	Required	Text to parse as a JSON object. The string must conform to the JSON standard.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
reviver	Function	Optional	Specifies how to transform the parsed value before it is returned	Version 2015 Release 2

## Syntax

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

```
...
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)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Converts a JavaScript object or value to a JSON string  For more information about JSON object format, see <a href="http://www.ietf.org/rfc/rfc4627.txt">http://www.ietf.org/rfc/rfc4627.txt</a> .
<b>Returns</b>	JSON string
<b>Supported Script Types</b>	All script types
<b>Governance</b>	None
<b>Global object</b>	JSON object
<b>Since</b>	Version 2015 Release 2

## Parameters

Parameter	Type	Required / Optional	Description	Since
value	Object	Required	The value to convert to a JSON string	Version 2015 Release 2
replacer	Function	Optional	Function that changes the behavior of the stringification process	Version 2015 Release 2
space	Object	Optional	A string or number that is used to insert white space in the output JSON string for readability	Version 2015 Release 2

## Syntax

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

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

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

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

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

## Promise Object



**Note:** The content in this help topic pertains to SuiteScript 2.0.

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/>.

## 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.

```

/**
 * @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.

```

/**
 * @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.

```

/**
 * @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.

Module	Description
<a href="#">N/action Module</a>	Load the N/action module APIs to execute business logic to update the state of a record. Action APIs emulate NetSuite UI buttons.
<a href="#">N/auth Module</a>	Load the auth module when you want to change your NetSuite login credentials.
<a href="#">N/cache Module</a>	Load the cache module to enable the caching of needed data and improve performance.
<a href="#">N/certificateControl Module</a>	The 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 <a href="#">Digital Signing</a> and <a href="#">Uploading Digital Certificates</a> .
<a href="#">N/commerce Modules</a>	Use modules in the N/commerce namespace to access different assets in the web store context, such as items and shopping cart. The modules within the N/commerce namespace are supported by the latest version of SuiteCommerce and by SuiteCommerce Advanced 2019.2 onwards.
<a href="#">N/config Module</a>	Load the config module when you want to access NetSuite configuration settings. See <a href="#">config.Type</a> for a list of supported configuration pages.
<a href="#">N/crypto Module</a>	Load the crypto module to work with hashing, hash-based message authentication (hmac), and symmetrical encryption. You can access a set of wrappers for OpenSSL's hash, hmac, cipher, and decipher methods.
<a href="#">N/crypto/certificate Module</a>	Load the 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.
<a href="#">N/currency Module</a>	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.
<a href="#">N/currentRecord Module</a>	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.
<a href="#">N/email Module</a>	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.

Module	Description
N/encode Module	Load the encode module when you want to convert a string to another type of encoding. See <a href="#">encode.Encoding</a> for a list of supported character set encoding.
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/file Module	Load the file module to work with files in NetSuite.
N/format Module	Load the format module to convert strings into a specified format and to parse formatted data into strings.
N/format/i18n Module	Load the format/i18n module to format currency.
N/http Module	Load the http module to make http calls. All HTTP content types are supported.
N/https Module	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.
N/https/clientCertificate Module	Load the clientCertificate module to send SSL requests with a digital certificate.
N/log Module	Load the log module when you want to access methods for logging script execution details. Module members are also supported by the global <a href="#">log Object</a> .
N/piremoval Module	Load the N/piremoval module to remove personal information (PI) from system notes, workflow history, and specific field values.
N/plugin Module	Load the plugin module to load custom plug-in implementations.
N/portlet Module	Load the portlet module when you want to resize or refresh a form portlet.
N/query Module	Load the query module to create and run searches using the SuiteAnalytics Workbook query engine.
N/record Module	Load the record module to work with NetSuite records.
N/redirect Module	Load the redirect module when you want to redirect users to one of the following: <ul style="list-style-type: none"><li>■ URL</li><li>■ Suitelet</li><li>■ Record</li><li>■ Task link</li><li>■ Saved search</li><li>■ Unsaved search</li></ul>
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

Module	Description
	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/task/accounting/recognition Module	Load the task/accounting/recognition module to merge revenue arrangements or revenue elements. The task/accounting/recognition module lets you combine revenue arrangements or revenue elements from multiple sources to represent a single contract for revenue allocation and recognition.
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 create 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 <a href="#">util Object</a> .
N/workflow Module	Load the workflow module to initiate new workflow instances or trigger existing workflow instances.
N/xml Module	Load the xml module to validate, parse, read, and modify XML documents.

## N/action Module



**Note:** This content in this topic applies to SuiteScript 2.0.

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](#).



**Note:** For supported script types, see individual member topics listed below.

- [N/action Module Members](#)
- [Action Object Members](#)
- [N/action Module Script Samples](#)

## N/action Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">action.Action</a>	Object	Client and server-side scripts	Encapsulates a NetSuite record action.
	Plain JavaScript Object	Object	Client and server-side scripts	A plain JavaScript object of actions available for a record type.
Method	<a href="#">action.execute(options)</a>	Object	Client and server-side scripts	Executes the record action and returns action results in an object.
	<a href="#">action.execute.promise(options)</a>	Promise	Client scripts	Asynchronously executes the record action and returns the action results in an object.
	<a href="#">action.executeBulk(options)</a>	string	Client and server-side scripts	Executes an asynchronous bulk record action and returns its task ID for later status inquiry.
	<a href="#">action.find(options)</a>	Object	Client and server-side scripts	Returns a plain JavaScript object of available record actions for the given record type.
	<a href="#">action.find.promise(options)</a>	Promise	Client scripts	Asynchronously returns a plain JavaScript object of available record actions for the given record type.
	<a href="#">action.get(options)</a>	<a href="#">action.Action</a>	Client and server-side scripts	Returns an executable record action for the given record type.
	<a href="#">action.get.promise(options)</a>	Promise	Client scripts	Asynchronously returns an executable record action for the given record type.

## Action Object Members

The following members are called on [action.Action](#).

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

## 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](#).



**Note:** This sample script uses the **require** function so that you can copy it into the SuiteScript Debugger and test it. You must use the **define** function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

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

```
/**
```

```

* @NApiVersion 2.x
*/
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

```

**i Note:** This sample script uses the **require** function so that you can copy it into the SuiteScript Debugger and test it. You must use the **define** function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

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

```

/**
* @NApiVersion 2.x
* @NScriptType ClientScript
*/
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

```

**i Note:** This sample script uses the **require** function so that you can copy it into the SuiteScript Debugger and test it. You must use the **define** function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to use the `action.executeBulk(options)`.

```

/**
 * @NApiVersion 2.x
 */

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 sample are very similar and can be generated programmatically 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: searchResults.map(function(v) {
        return { recordId: v, foo: "bar", name: "John Doe" };
    })
});

// 2b) Bulk execute the specified action on a provided list of record IDs.
// This time with homogenous parameters. Equivalent to the previous sample.
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 recordIds in hand, we only provide the callback
// that will later be used to transform a recordId 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 };
        }
    });

// 8) Get status of a bulk action execution.
var res = action.getBulkStatus({taskId: handle}); // returns a RecordActionTaskStatus object
log.debug(res.status);
});

```

## action.Action



**Note:** This content in this topic applies to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a NetSuite record action.  This object is returned by the <a href="#">action.get(options)</a> and <a href="#">action.find(options)</a> methods.
<b>Supported Script Types</b>	Client and server scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/action Module</a>
<b>Methods and Properties</b>	<a href="#">Action Object Members</a>
<b>Since</b>	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](#).

```

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

```

## Action(options)



**Note:** This content in this topic applies to SuiteScript 2.0.

<b>Method Description</b>	Executes the action and returns the action result in a plain JavaScript object.  The action result is returned in an object. The <b>response</b> property of the results object shows the action result. If the action fails, it is listed in the results object's <b>notifications</b> property. If the action executes successfully, the <b>notifications</b> property is usually empty.  If the Action object is qualified (it is a result of an <code>action.get()</code> or <code>action.find()</code> call that provides the <code>recordId</code> ), then it is not required to provide a <code>recordId</code> and the
---------------------------	--

	<p><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.</p> <p><b>Note:</b> Replace Action with the name of the action you are executing.</p>
Returns	Object
Supported Script Types	Client and server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Module	N/action Module
Parent Object	<code>action.Action</code>
Sibling Object Members	<a href="#">Action Object Members</a>
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`.

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

## Errors

## Syntax

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	A required parameter is missing.

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

## Action.promise(options)

**Note:** This content in this topic applies to SuiteScript 2.0.

<b>Method Description</b>	<p>Executes the action asynchronously and returns the action result in a plain JavaScript object. The action result is returned in an object. The <b>response</b> property of the results object shows the action result. If the action fails, it is listed in the results object's <b>notifications</b> property. If the action executes successfully, the <b>notifications</b> property is usually empty.</p> <p>If the Action object is qualified (it is a result of an <code>action.get()</code> or <code>action.find()</code> call that provides the <code>recordId</code>), 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.</p> <p><b>Note:</b> Replace Action with the name of the action you are executing.</p> <p><b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">Action(options)</a>. For more information on promises, see <a href="#">Promise Object</a>.</p>
<b>Returns</b>	<a href="#">Promise Object</a>
<b>Supported Script Types</b>	Client scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/action Module</a>
<b>Parent Object</b>	<a href="#">action.Action</a>
<b>Sibling Object Members</b>	<a href="#">Action Object Members</a>
<b>Since</b>	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`.

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

## Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	A required parameter is missing.

## Syntax



**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

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

## Action.execute(options)



**Note:** This content in this topic applies to SuiteScript 2.0.

<b>Method Description</b>	Executes the action and returns the action result in a object.  The <b>response</b> property of the result object holds the actual response returned by the action implementation. The <b>notifications</b> 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 <b>notifications</b> property is usually empty.  If the Action object is qualified (it is a result of an <code>action.get()</code> or <code>action.find()</code> call that provides the recordId), then it is not required to provide a <b>recordId</b> and the <b>options.params.recordId</b> parameter is optional. If <b>options.params.recordId</b> is provided during execution, it takes precedence over the <b>recordId</b> stored in the Action object.
<b>Returns</b>	Object
<b>Supported Script Types</b>	Client and server scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/action Module
<b>Parent Object</b>	<a href="#">action.Action</a>
<b>Sibling Object Members</b>	<a href="#">Action Object Members</a>
<b>Since</b>	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.

Parameter	Type	Required / Optional	Description
<b>options.Object</b>	Object	required or optional	The parameters that need to be provided depend on the action implementation. See the action help.
<b>options.params.recordId</b>	string	required or optional	The record instance ID of the record on which the action is to be performed.

Parameter	Type	Required / Optional	Description
			This is the NetSuite record internal ID.

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required 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/action Module Script Samples](#).

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

## Action.execute.promise(options)

 **Note:** This content in this topic applies to SuiteScript 2.0.

<b>Method Description</b>	Executes the action asynchronously and returns the action result in a plain JavaScript object.  The action result is returned in an object. The <b>response</b> property of the results object shows the action result. If the action fails, it is listed in the results object's <b>notifications</b> property. If the action executes successfully, the <b>notifications</b> property is usually empty.  If the Action object is qualified (it is a result of an <code>action.get()</code> or <code>action.find()</code> call that provides the <code>recordId</code> ), 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.
	 <b>Note:</b> The parameters and errors thrown for this method are the same as those for <code>Action.execute(options)</code> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Supported Script Types</b>	Client scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/action Module
<b>Parent Object</b>	<code>action.Action</code>
<b>Sibling Object Members</b>	<a href="#">Action Object Members</a>
<b>Since</b>	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`.

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

## Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	A required parameter is missing.

## Syntax

**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

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

## Action.executeBulk(options)

**Note:** This content in this topic applies to SuiteScript 2.0.

<b>Method Description</b>	Executes an asynchronous bulk record action and returns its task ID for status queries with <a href="#">action.getBulkStatus(options)</a> .
<b>Returns</b>	string
<b>Supported Script Types</b>	Client and server-side scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	50 usage units
<b>Module</b>	<a href="#">N/action Module</a>
<b>Parent Object</b>	<a href="#">action.Action</a>
<b>Sibling Object Members</b>	<a href="#">Action Object Members</a>

Since	2019.1
-------	--------

## Parameters

<b>i Note:</b>	The <code>options.params</code> 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 <code>options.recordId</code> , unless the action object is qualified. An action object is qualified if it is the result of an <code>action.get()</code> or <code>action.find()</code> call that provides the <code>recordId</code> .
----------------	---

Parameter	Type	Required / Optional	Description
<code>options.params</code>	array	optional	<p>The <code>options.params</code> parameter is mutually exclusive to <code>options.condition</code> and <code>options.paramCallback</code>.</p> <p>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:</p> <pre>{recordId: 1, someParam: 'example1', otherParam: 'example2'}</pre>
<code>options.condition</code>	string	optional	<p>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.</p> <p>The <code>action.ALL_QUALIFIED_INSTANCES</code> condition only works correctly if the author of the record action has implemented the <code>findInstances</code> method of the <code>RecordActionQualifier</code> interface. An example of such action is <code>approve</code> on the timebill and timesheet records.</p>
<code>options.paramCallback</code>	string	optional	the name of the function that takes a record ID and returns the parameter object for the specified record ID.

## Errors

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

## Syntax

<b>Important:</b>	The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/action Module Script Samples</a> .
-------------------	--

```
// 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'
        }
    ]
});
...
// Add additional code

```

## action.getBulkStatus(options)

**i Note:** This content in this topic applies to SuiteScript 2.0.

<b>Method Description</b>	Returns the current status of <a href="#">action.executeBulk(options)</a> for the specified task ID. The bulk execution status is returned in a status object.
<b>Returns</b>	<a href="#">RecordActionTaskStatus Object Members</a>
<b>Supported Script Types</b>	Client and server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/action Module</a>
<b>Sibling Object Members</b>	<a href="#">N/action Module Members</a>
<b>Since</b>	2019.1

## Parameters

Parameter	Type	Required / Optional	Description
options.taskId	string	required	The task ID returned by a previous <a href="#">action.executeBulk(options)</a> call.

## Syntax

**⚠ 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
```

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

## Action.description

 **Note:** This content in this topic applies to SuiteScript 2.0.

<b>Property Description</b>	The action description.
<b>Type</b>	string
<b>Supported Script Types</b>	Client and server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/action Module</a>
<b>Sibling Object Members</b>	<a href="#">Action Object Members</a>
<b>Since</b>	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 Sample](#).

```
// Add additional code
...
var description = action.description; // get the action description
...
// Add additional code
```

## Action.id

 **Note:** This content in this topic applies to SuiteScript 2.0.

<b>Property Description</b>	The ID of the action. For a list of action IDs, see the help topic <a href="#">Supported Record Actions</a> .
<b>Type</b>	string
<b>Supported Script Types</b>	Client and server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/action Module</a>

<b>Sibling Object Members</b>	Action Object Members
<b>Since</b>	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](#).

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

## Action.label

 **Note:** This content in this topic applies to SuiteScript 2.0.

<b>Property Description</b>	The action label.
<b>Type</b>	string
<b>Supported Script Types</b>	Client and server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/action Module
<b>Sibling Object Members</b>	Action Object Members
<b>Since</b>	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](#).

```
// Add additional code
...
var label = action.label; // get the action label
...
// Add additional code
```

## Action.parameters

 **Note:** This content in this topic applies to SuiteScript 2.0.

<b>Property Description</b>	The action parameters.
-----------------------------	------------------------

<b>Type</b>	Object
<b>Supported Script Types</b>	Client and server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/action Module
<b>Sibling Object Members</b>	Action Object Members
<b>Since</b>	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](#).

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

## Action.recordType

 **Note:** This content in this topic applies to SuiteScript 2.0.

<b>Property Description</b>	The type of the record on which the action is to be performed. For a list of record types, see <a href="#">record.Type</a> .
<b>Type</b>	string
<b>Supported Script Types</b>	Client and server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/action Module
<b>Sibling Object Members</b>	Action Object Members
<b>Since</b>	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](#).

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

## action.execute(options)



**Note:** This content in this topic applies to SuiteScript 2.0.

<b>Method Description</b>	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 <b>notifications</b> property. If the action executes successfully, the <b>notifications</b> property is usually empty.
<b>Returns</b>	Object
<b>Supported Script Types</b>	Client and server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/action Module</a>
<b>Sibling Object Members</b>	<a href="#">N/action Module Members</a>
<b>Since</b>	2018.2

## Parameters



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

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

## Errors

Error Code	Thrown If
SSS_INVALID_RECORD_TYPE	The specified record type is invalid.
SSS_MISSING_REQD_ARGUMENT	A required parameter is missing.
SSS_INVALID_ACTION_ID	The specified action does not exist on the specified record type. - or - The action exists, but cannot be executed on the specified record instance.
RECORD_DOES_NOT_EXIST	The specified record instance does not exist.

## Syntax



**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](#) and [Revenue Arrangement Record Actions](#).

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

## action.execute.promise(options)



**Note:** This content in this topic applies to SuiteScript 2.0.

<b>Method Description</b>	Executes the record action asynchronously.  If the action fails, it is listed in the results object's <b>notifications</b> property. If the action executes successfully, the <b>notifications</b> property is usually empty.
<b>Returns</b>	Promise Object
<b>Supported Script Types</b>	Client scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/action Module</a>
<b>Sibling Object Members</b>	<a href="#">N/action Module Members</a>
<b>Since</b>	2018.2

## Parameters



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

Parameter	Type	Required / Optional	Description
options.recordType	string	required	The record type.  For a list of record types, see <a href="#">record.Type</a> .
options.id	string	required	The action ID.  For a list of action IDs, see the help topic <a href="#">Supported Record Actions</a> .

Parameter	Type	Required / Optional	Description
options.params	Object	required	Action arguments.
options.params.recordId	string	required	The record instance ID. This is the NetSuite record internal ID.

## Errors

Error Code	Thrown If
SSS_INVALID_RECORD_TYPE	The specified record type is invalid.
SSS_MISSING_REQD_ARGUMENT	A required parameter is missing.
SSS_INVALID_ACTION_ID	The specified action does not exist on the specified record type. - or - The action exists, but cannot be executed on the specified record instance.
RECORD_DOES_NOT_EXIST	The specified record instance does not exist.

## Syntax



**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// 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)



**Note:** This content in this topic applies to SuiteScript 2.0.

<b>Method Description</b>	Executes an asynchronous bulk record action and returns its task ID for status queries with <a href="#">action.getBulkStatus(options)</a> . The <code>options.params</code> parameter is mutually exclusive to <code>options.condition</code> and <code>options.paramCallback</code> .
<b>Returns</b>	string
<b>Supported Script Types</b>	Client and server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	50 usage units

<b>Module</b>	N/action Module
<b>Sibling Object Members</b>	N/action Module Members
<b>Since</b>	2019.1

## 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`.

Parameter	Type	Required / Optional	Description
<code>options.recordType</code>	string	required	The record type. For a list of record types, see <a href="#">record.Type</a> .
<code>options.id</code>	string	required	The action ID.
<code>options.params</code>	array	optional	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:  <pre>{recordId: 1, someParam: 'example1', otherParam: 'example2'}</pre> The <code>recordId</code> parameter is always mandatory, other parameters are optional and are specific to the particular action.
<code>options.condition</code>	string	optional	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.  The <code>action.ALL_QUALIFIED_INSTANCES</code> condition only works correctly if the author of the record action has implemented the <code>findInstances</code> method of the <code>RecordActionQualifier</code> interface. An example of such action is <code>approve</code> on the timebill and timesheet records.
<code>options.paramCallback</code>	string	optional	Function that takes record ID and returns the parameter object for the specified record ID.

## Errors

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

## Syntax

**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 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)

**Note:** This content in this topic applies to SuiteScript 2.0.

<b>Method Description</b>	Performs a search for available record actions. If only the <code>recordType</code> parameter is specified, all actions available for the record type are returned. If the <code>recordId</code> parameter is also specified, then only actions that qualify for execution on the given record instance are returned. If the <code>id</code> 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 <code>action.Action</code> objects. If there are no available actions for the specified record type, an empty object is returned.  If the <code>recordId</code> is specified in this call, the actions that are found are considered qualified. You do not have to provide the <code>recordId</code> to execute a qualified action.
<b>Returns</b>	Object
<b>Supported Script Types</b>	Client and server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/action Module</a>
<b>Sibling Object Members</b>	<a href="#">N/action Module Members</a>
<b>Since</b>	2018.2

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.recordType</code>	string	required	The record type.

Parameter	Type	Required / Optional	Description
			For a list of record types, see <a href="#">record.Type</a> .
<code>options.recordId</code>	string	optional	The record instance ID.
<code>options.id</code>	string	optional	The action ID.

## Errors

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

## Syntax



**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)



**Note:** This content in this topic applies to SuiteScript 2.0.

<b>Method Description</b>	<p>Performs a search for available record actions asynchronously. If only the <code>recordType</code> parameter is specified, all actions available for the record type are returned. If the <code>recordId</code> parameter is also specified, then only actions that qualify for execution on the given record instance are returned. If the <code>id</code> parameter is specified, the only the action with the specified action ID is returned.</p> <p>This method returns a plain JavaScript object of NetSuite record actions available for the record type. The object contains one or more <code>action.Action</code> objects. If there are no available actions for the specified record type, an empty object is returned.</p> <p>If the <code>recordId</code> is specified in this call, the actions that are found are considered qualified. You do not have to provide the <code>recordId</code> to execute a qualified action.</p>
---------------------------	---

	<b>Note:</b> The parameters and errors thrown for this method are the same as those for <code>action.find(options)</code> . For more information on promises, see <a href="#">Promise Object</a> .
Returns	Promise Object
Supported Script Types	Client scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Module	N/action Module
Sibling Object Members	N/action Module Members
Since	2018.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description
<code>options.recordType</code>	string	required	The record type. For a list of record types, see <a href="#">record.Type</a> .
<code>options.recordId</code>	string	optional	The record instance ID.
<code>options.id</code>	string	optional	The action ID.

## Errors

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

## Syntax

<b>Important:</b> The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see <a href="#">Promise Object</a> .
---

```
// 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)

 **Note:** This content in this topic applies to SuiteScript 2.0.

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

## Parameters

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

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

## Errors

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

## Syntax



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

## action.get.promise(options)



**Note:** This content in this topic applies to SuiteScript 2.0.

<b>Method Description</b>	Returns an executable record action for the specified record type asynchronously. If the <b>recordId</b> parameter is specified, the action object is returned only if the specified action can be executed on the specified record instance.
	<p><b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">action.get(options)</a>. For more information on promises, see <a href="#">Promise Object</a>.</p>
<b>Returns</b>	<a href="#">Promise Object</a>
<b>Supported Script Types</b>	Client scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/action Module</a>
<b>Sibling Object Members</b>	<a href="#">N/action Module Members</a>
<b>Since</b>	2018.2

## Parameters



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

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

## Errors

Error Code	Thrown If
SSS_INVALID_RECORD_TYPE	The specified record type is invalid.
SSS_MISSING_REQD_ARGUMENT	A required parameter is missing.
SSS_INVALID_ACTION_ID	The specified action does not exist on the specified record type. – or – The action exists, but cannot be executed on the specified record instance.
RECORD_DOES_NOT_EXIST	The specified record instance does not exist.

## Syntax



**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	auth.changeEmail (options)	void	Server scripts	Changes the current user's NetSuite email address (user name).
	auth.changePassword (options)	void	Server scripts	Changes the current user's NetSuite password.

## N/auth Module Script Sample

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to change the currently logged-in user's NetSuite email address and password.

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

**Important:** The value used in this sample for the password and email fields are placeholders. Before using this sample, replace the password and email field values with valid values from your NetSuite account. If you run a script with an invalid value, an error may occur.

```
/**  
 * @NApiVersion 2.x  
 */  
  
//The following script changes the currently logged-in user's NetSuite email address and password.  
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)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Changes the current user's NetSuite email address (user name).
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 usage units
<b>Module</b>	<a href="#">N/auth Module</a>
<b>Since</b>	2015.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.			
<b>Parameter</b>	<b>Type</b>	<b>Required / Optional</b>	<b>Description</b>
<code>options.password</code>	string	required	The logged in user's current NetSuite password.
<code>options.newEmail</code>	string	required	The logged in user's NetSuite new email address.
<code>options.onlyThisAccount</code>	boolean	optional	Determines which accounts the email address is changed for. If set to <code>true</code> , the email address change is applied only to roles within the current account. If set to <code>false</code> , the email address change is applied to all accounts and roles.  The default value is <code>true</code> .

## Errors

Error Code	Message	Thrown If
INVALID_PSWD		The <code>options.password</code> does not conform to the password rules.  See the help topic <a href="#">Creating a Strong Password</a> for information on valid passwords.
INVALID_EMAIL		The <code>options.newEmail</code> is invalid.

## Syntax

**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](#).

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

## auth.changePassword(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Changes the current user's NetSuite password.  See the help topic <a href="#">Creating a Strong Password</a> for information on valid passwords.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server-side scripts

	For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 usage units
<b>Module</b>	<a href="#">N/auth Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.currentPassword	string	required	The logged in user's current NetSuite password
options.newPassword	string	required	The logged in user's new NetSuite password. See the help topic <a href="#">Creating a Strong Password</a> for information on valid passwords.

## Errors

Error Code	Message	Thrown If
INVALID_PSWD		The <code>options.password</code> does not conform to the password rules. See the help topic <a href="#">Creating a Strong Password</a> for information on valid passwords.
INVALID_EMAIL		The <code>options.newEmail</code> is invalid.

## Syntax

 **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](#).

```
//Add additional code
...
auth.changePassword({
    currentPassword: 'mycurrentPWD',
    newPassword: 'mynewPWD_2*'
});
...
//Add additional code
```

## N/cache Module

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

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 (1) to the current script only, (2) to all server-side scripts in the current bundle, or (3) 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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">cache.Cache</a>	Object	Server-side scripts	Encapsulates a segment of memory that can be used to temporarily store data on a short-term basis.
Method	<a href="#">cache.getCache(options)</a>	<a href="#">cache.Cache</a>	Server-side scripts	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.
Enum	<a href="#">cache.Scope</a>	enum	Server-side scripts	An enum used to populate the Cache.scope property.

## Cache Object Members

The following members are called on [cache.Cache](#).

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Method	<a href="#">Cache.get(options)</a>	string	Server-side scripts	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.
	<a href="#">Cache.put(options)</a>	string	Server-side scripts	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.
	<a href="#">Cache.remove(options)</a>	string	Server-side scripts	Removes a value from the cache.
Property	<a href="#">Cache.name</a>	string	Server-side scripts	A label that identifies the cache.
	<a href="#">Cache.scope</a>	string	Server-side scripts	A value that describes the availability of the cache. A cache can be made

Member Type	Name	Return Type/ Value Type	Supported Script Types	Description
				available to the current script only, to all scripts in the current bundle, or to all scripts in your NetSuite account.

## N/cache Module Script Sample

**Note:** This script sample uses the `define` function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the `require` function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample shows how to retrieve the name of a city based on a ZIP code using a Suitelet. To speed up 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 loader function is a custom module called `zipCodeDatabaseLoader` (shown in the second script sample). It loads a CSV file and uses it to find the requested value.

**Note:** This sample depends on a CSV file that must exist before the script is run. The sample CSV file is available [here](#).

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

//This script retrieves the name of a city based on a ZIP code, using a cache.
define(['N/cache', '/SuiteScripts/zipCodes/ca/zipToCityIndexCacheLoader'],
    function (cache, lib){
        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');
}
return {
    onRequest: onRequest
};
});
});

```

The following custom module returns the loader function used in the preceding script sample. The loader function shows how to use a CSV file to retrieve a value that was missing from a cache. This custom module 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
 */

//This is a loader function that uses a CSV file to retrieve a value that was missing from a cache.
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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	A segment of memory that can be used to temporarily store data (on a short term basis) needed by one script, by all scripts in a bundle, or by all scripts in the NetSuite account.  This object is returned by <a href="#">cache.getCache(options)</a> .
<b>Supported Script Types</b>	Server-side scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/cache Module</a>
<b>Methods and Properties</b>	<a href="#">Cache Object Members</a>
<b>Since</b>	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](#).

```
//Add additional code
...
//myCache in the following statement will be a cache.Cache object as returned from cache.getCache
var myCache = cache.getCache({
    name: 'temporaryCache',
    scope: cache.Scope.PRIVATE
});
...
//Add additional code
```

## Cache.get(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Retrieves a string value from the cache. The value retrieved is identified by a key that you pass by using the <code>options.key</code> parameter. If a requested value is not present in the cache, the system calls the function identified by the <code>options.loader</code> parameter. This user-defined function should provide logic for retrieving a value that is not in the cache. For an example, see <a href="#">N/cache Module Script Sample</a> .
<b>Returns</b>	String or null
<b>Supported Script Types</b>	Server-side scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	1 unit if the value is present in the cache; 2 units if the loader function is used

<b>Module</b>	N/cache Module
<b>Parent Object</b>	cache.Cache
<b>Sibling Object Members</b>	Cache Object Members
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.key	string	required	A string that identifies the value to be retrieved from the cache. This value cannot be null.
options.loader	function	optional, but strongly recommended	<p>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 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 <a href="#">N/cache Module Script Sample</a>.</p> <p>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.</p> <p>When no loader is specified and a value is missing from the cache, the system returns null.</p>
options.ttl	number	optional	<p>The maximum duration, in seconds, that a value retrieved by the loader can remain in the cache. Note that the value may be removed from the cache before the <b>ttl</b> limit is reached.</p> <p>The minimum value is 300 (five minutes) and there is no maximum. The default <b>ttl</b> value is no limit.</p> <p> <b>Important:</b> A cached value is not guaranteed to stay in the cache for the full duration of the <b>ttl</b> value. The <b>ttl</b> value represents the maximum time that the cached value may be stored.</p>

## Syntax

 **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
...
var myCache = cache.getCache({
  name: 'temporaryCache',
  scope: cache.Scope.PRIVATE
})
```

```

});  
  

var myValue = myCache.get({  

    key: 'keyText',  

    loader: loader,  

    ttl: 18000  

});  

...
//Add additional code

```

## Cache.put(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Puts a value into the cache.   <b>Note:</b> You can also put a value in a cache by using the <a href="#">Cache.get(options)</a> 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 <a href="#">N/cache Module Script Sample</a>
<b>Returns</b>	void
<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	1 unit
<b>Module</b>	N/cache Module
<b>Parent Object</b>	<a href="#">cache.Cache</a>
<b>Sibling Object Members</b>	<a href="#">Cache Object Members</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.key	string	required	The identifier of the value that is being cached. The maximum size of the cache key is 4 kilobytes.
options.value	string	required	The value to place in the cache. If the value submitted is not a string, the system uses <code>JSON.stringify()</code> to convert the value before it is placed in the cache. The maximum size of the value is 500KB.
options.ttl	number	optional	The maximum duration, in seconds, that the value may remain in the cache. Note that the value may be removed before the <code>ttl</code> limit is reached.

Parameter	Type	Required / Optional	Description
			<p>The minimum value is 300 (five minutes) and there is no maximum. The default <code>ttl</code> value is no limit.</p> <div style="border: 1px solid #f0e68c; padding: 10px; margin-top: 10px;"> <span style="color: yellow;"> <b>Important:</b></span> A cached value is not guaranteed to stay in the cache for the full duration of the <code>ttl</code> value. The <code>ttl</code> 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 <a href="#">Cache.get(options)</a> method and options.loader parameter to set and retrieve data.         </div>

## Syntax

 **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
...
var myCache = cache.getCache({
    name: 'temporaryCache',
    scope: cache.Scope.PRIVATE
});
myCache.put({
    key: 'keyText',
    value: 'valueText',
    ttl: 300
});
...
//Add additional code
```

## Cache.remove(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Removes a value from the cache.
<b>Returns</b>	void
<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	1 unit
<b>Module</b>	<a href="#">N/cache Module</a>
<b>Parent Object</b>	<code>cache.Cache</code>
<b>Sibling Object Members</b>	<a href="#">Cache Object Members</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.key	string	required	The identifier of the value that is being removed.

## Syntax

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

## Cache.name

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	A label that identifies a cache.
<b>Type</b>	string
<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/cache Module
<b>Parent Object</b>	cache.Cache
<b>Sibling Object Members</b>	Cache Object Members
<b>Since</b>	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](#)

```
//Add additional code
```

```

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

```

## Cache.scope

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	A label that describes the availability of the cache to other scripts.
<b>Type</b>	<a href="#">cache.Scope</a>
<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/cache Module</a>
<b>Parent Object</b>	<a href="#">cache.Cache</a>
<b>Sibling Object Members</b>	<a href="#">Cache Object Members</a>
<b>Since</b>	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](#).

```

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

```

## cache.getCache(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	<a href="#">cache.Cache</a>
<b>Supported Script Types</b>	All server-side scripts

	For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	n/a
<b>Module</b>	<a href="#">N/cache Module</a>
<b>Sibling Module Members</b>	<a href="#">N/cache Module Members</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.name</code>	string	required	A label identifies the cache to be retrieved or to be created. The maximum size of the cache name is 1 kilobyte.
<code>options.scope</code>	string	optional, if you do not set a value, the default value PRIVATE is used	This value is set with the <code>cache.Scope</code> 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.

## Syntax

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

## cache.Scope

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds string values that describe the availability of the cache. This enum is used to set the value of the <code>Cache.scope</code> property.
<b>Type</b>	 <b>Note:</b> 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.

<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/cache Module</a>
<b>Sibling Module Members</b>	<a href="#">N/cache Module Members</a>
<b>Since</b>	2016.2

## Values

Value	Description
PRIVATE	The cache is available only to the current script. This value is the default.
PROTECTED	The cache is available only to some scripts, as follows: <ul style="list-style-type: none"> <li>■ If the script is part of a bundle, the cache is available to all scripts in the same bundle.</li> <li>■ If the script is not in a bundle, the cache is available to all scripts not in any bundle.</li> </ul>
PUBLIC	The cache is available to any script in the NetSuite account.

## Syntax



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

## N/certificateControl Module



**Note:** The content in this help topic pertains to SuiteScript 2.0.

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, create, update, read and delete certificates records. 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](#).



**Important:** The certificate record holds information for a digital certificate, but it is not a standard NetSuite record and cannot be accessed with the N/record.

## N/certificateControl Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<code>certificateControl.Certificate</code>	object	server scripts	Encapsulates a digital certificate record.
Method	<code>certificateControl.findCertificates(options)</code>	object	server scripts	Returns metadata about the certificate(s).
	<code>certificateControl.findUsages(options)</code>	object[]	server scripts	Returns an audit trail of how a certificate has been used. Includes operations performed with time stamps.
	<code>certificateControl.createCertificate(options)</code>	<code>certificateControl.Certificate</code>	server scripts	Creates a certificate record using a file from the File Cabinet. After saving with <code>Certificate.save()</code> , the certificate is accessible on the <b>Certificates</b> .
	<code>certificateControl.deleteCertificate(options)</code>	string	server scripts	Deletes a certificate record that has been uploaded to the <b>Certificates</b> list in the UI or created using <code>certificateControl.createCertificate(options)</code> and saved with <code>Certificate.save()</code> .
	<code>certificateControl.loadCertificate(options)</code>	<code>certificateControl.Certificate</code>	server scripts	Loads a certificate record that has been uploaded to the <b>Certificates</b> list in the UI or created using <code>certificateControl.createCertificate(options)</code> .
Enum	<code>certificateControl.Type</code>	enum	server scripts	Enum for certificate types. PFX, PEM, and P12 are supported types.
	<code>certificateControl.Operation</code>	enum	server scripts	Enum for searching the audit trail of certificates with <code>certificateControl.findUsages(options)</code> .
	<code>certificateControl.Operator</code>	enum	server scripts	Enum for searching for certificate records with <code>certificateControl.findCertificates(options)</code> .

## Certificate Object Members

The following members are called on the `certificateControl.Certificate` object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<code>Certificate.save()</code>	object containing the script ID of the new certificate record	server scripts	Saves a certificate record.
Property	<code>Certificate.description</code>	string	server scripts	Describes the certificate record.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	Certificate.file	File Object Members object	server scripts	Includes the properties of the file uploaded to create the certificate.
	Certificate.name	string	server scripts	The name of the certificate record.
	Certificate.monthReminder	boolean true false	server scripts	Indicates the setting of the <b>Month</b> box for <b>Expiration Reminders</b> on the certificate record.
	Certificate.notifications	number[]	server scripts	The internal IDs of the employees selected in the <b>Copy Employees</b> field on the certificate record.
	Certificate.password	string (write-only)	server scripts	The password for the digital certificate. You can create a GUID using <a href="#">Form.addSecretKeyField (options)</a> .
	Certificate.restrictions	number[]	server scripts	The internal IDs of the employees selected in the Restrict to Employees field of the certificate record.
	Certificate.scriptId	string	server scripts	The ID of the certificate record.
	Certificate.subsidiaries	number[]	server scripts	The internal IDs of the subsidiaries associated with the certificate record.
	Certificate.threeMonths Reminder	boolean true false	server scripts	Indicates the setting of the <b>3 Months</b> box for <b>Expiration Reminders</b> on the certificate record.
	Certificate.weekReminder	boolean true false	server scripts	Indicates the setting of the <b>Week</b> box for <b>Expiration Reminders</b> on the certificate record.

## N/certificateControl Module Script Samples

### Example 1

**i Note:** This sample script uses the **require** function so that you can copy it into the SuiteScript Debugger and test it. You must use the **define** function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

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

```
/*
 * @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
});

```

## Example 2

**i Note:** This sample script uses the **require** function so that you can copy it into the SuiteScript Debugger and test it. You must use the **define** function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to find the audit trail of POST operations for the certificate record with ID 'custcertificate\_china'.

```

/**
 * @NApiVersion 2.x
 */

require(['N/certificateControl'], function(cc){
    var usages = cc.findUsages({
        id: 'custcertificate_china',
        operation: cc.Operation.POST
    });
})

```

## Example 3

**i Note:** This sample script uses the **require** function so that you can copy it into the SuiteScript Debugger and test it. You must use the **define** function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to create a file object by loading a file from the File Cabinet. It then creates the options needed for the **certificateControl.createCertificate(options)** method and creates and saves the certificate record. The certificate record is then loaded again, edited to change the file, and saved again.

```

/**
 * @NApiVersion 2.x
 */

require(['N/certificateControl','N/file'],function(cc, file){
    var fileObj = file.load({
        id: 'SuiteScripts/dsa.p12'
    });
    var options = {
        file : fileObj,
        password : '022b490ad4334c7e86a8304f937ec68f',
        name : 'testCert',
        description : 'testDescription',
    }
})

```

```

       scriptId : '_testid',
subsidiaries : [1,3],
weekReminder : false,
monthReminder : true,
threeMonthsReminder : false
};

var newCertificate = cc.createCertificate(options);
newCertificate.save();

var loadedCertificate = cc.loadCertificate({
    scriptId : 'custcertificate_testid'
});
fileObj = file.load({
    id: 'SuiteScripts/ecdsa.p12'
});
loadedCertificate.file = fileObj;
loadedCertificate.password = '022b490ad4334c7e86a8304f937ec68f';
loadedCertificate.save();
})

```

## Example 4

**i Note:** This sample script uses the **require** function so that you can copy it into the SuiteScript Debugger and test it. You must use the **define** function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to find an existing certificate record and use it in an operation.

```

/**
 * @NApiVersion 2.x
 */

require(['N/certificateControl','N/https/clientCertificate'],function(cc, cert){
    var yodlee = cc.findCertificates({
        name: 'Yodlee',
        description: 'Yodlee certificate'
    });
    cert.post({certId:yodlee[0].id,url:<url>, body:<body>, headers:<headers>}
    );
})

```

## Example 5

**i Note:** This sample script uses the **require** function so that you can copy it into the SuiteScript Debugger and test it. You must use the **define** function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to generate a signature of a plaintext string and then verifies the signature using the same certificate.

```

/**
 * @NApiVersion 2.x
 */

```

```

require(['N/certificateControl', 'N/crypto/certificate'], function(cc, certificate){
    var signer = certificate.createSigner({
        certId: 'custcertificate_cert_1',
        algorithm: 'SHA256'
    });
    var result = signer.sign();
    var verifier = certificate.createVerifier({
        certId: 'custcertificate_cert_1',
        algorithm: 'SHA256'
    });
    verifier.update('test');
    verifier.verify(result);
    var res = cc.findUsages();
    ;
})

```

The **res** variable returns an array of information about the usage of the digital certificate, including the date of the action, the type of operation, such as **sign**, and the internal ID of the person who performed the action.

## Example 6

**i Note:** This sample script uses the **require** function so that you can copy it into the SuiteScript Debugger and test it. You must use the **define** function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample establishes a SFTP connection using an SSH key that has already been uploaded to NetSuite. It then creates, updates, loads, and deletes a certificate record to show the full CRUD operation. Replace the server URL with your correct URL.

For the SFTP connection, the public key corresponding to the private key in the certificate must be stored in the `.ssh/authorized_keys` file on the server.

```

/**
 * @NApiVersion 2.x
 */

require(['N/file', 'N/sftp', 'N/certificateControl'], function(file, sftp, certificateControl) {
    var certPath = 'yyy/certificates';
    var certName = 'apiclient_cert.p12';

    // Establish SFTP connection
    log.debug('Establishing secured SFTP connection...');
    var connection = sftp.createConnection({
        username: 'sftpuser',
        keyId: 'custkeysftp_nft_demo_key',
        url: 'my.sftp.server.com',
        port: 22,
        directory: 'inbound',
        hostKey: 'AAAAAB3NzaC1yc2EAAAQABAAQCAgYD1K41E9QnuYEgRRQChjrAM1+bTT95e71Xv0oQ60ywVQEiedhRqSMbPiCPPB
4pjpb0mpIQCCKug+3wxAQ6uNj3UM11zoGGmg86tyEJT6qGB0SsrQJzHtb3EG38BSrB00WEzOWeJ8E8Y0DT3oAj1Nrf8Ls3JbGobRF+0uwJD1
1SrFkYS3kWCV27NhBnaytGe7iLBgrJdNVlitNqkxZfK0NsAYCaJWQjQLtz+GFFn5zTbKKNsDa6s/YW7oAMMOI3Q5GQAqdXtKY728WvxYTji2FsYS/
KM6nbq/csTvZHWE0z2TQtB2H0IIofvEP/QvXwmqEnCeVPcNgRwdHWQf'
    });
}

```

```

    log.debug('Connection established!');
// ----

// Create new certificate
log.debug('Creating certificate...');
var certscriptId = '_' + (Math.random().toString(36).substring(2, 10));
var cert = certificateControl.createCertificate();
cert.name = 'Test Certificate China API';
cert.description = 'Test Certificate China created via API';
// custcertificate prefix will be added automatically
cert.scriptId = certscriptId;
log.debug('Downloading certificate file from SFTP...');
cert.file = connection.download({
    directory: certPath,
    filename: certName
});
log.debug('Successfully downloaded!');
//guid corresponding to the certificate's password';
var pwd = '022b490ad4334c7e86a8304f937ec68f',
cert.password = pwd;
cert.save();
/**/certscriptId = 'custcertificate' + certscriptId;
log.debug('Certificate "' + cert.name + '" successfully created with id "' + certscriptId + '"!');
// ----

// Rename certificate
log.debug('Renaming certificate...');
cert = certificateControl.loadCertificate({scriptId: certscriptId});
cert.name = 'Test Certificate China API TEMP';
cert.save();
// Verify new certificate name
/**/cert = certificateControl.loadCertificate({scriptId: certscriptId});
log.debug('Certificate successfully renamed to "' + cert.name + '"');
// ----

// Delete certificate
log.debug('Deleting certificate "' + cert.name + '"...');
certificateControl.deleteCertificate(certscriptId);
log.debug('Certificate deleted!');
// ----

// Load the deleted certificate
log.debug('Attempting to load the deleted certificate to invoke an error...');
try {
    cert = certificateControl.loadCertificate({scriptId: certscriptId});
}
catch (e) {
    log.error(e.message);
}
})

```

## certificateControl.Certificate



**Note:** The content in this help topic pertains to SuiteScript 2.0.

### Object Description

The certificate record, including file name and preferences.

<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/certificateControl Module</a>
<b>Methods and Properties</b>	<a href="#">Certificate Object Members</a>
<b>Since</b>	2019.2

## 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/certificateControl Module Script Samples](#).

```
// Add additional code
...
var loadedCertificate = cc.loadCertificate({
   scriptId : 'custcertificate_testid'
});
fileObj = file.load({
    id: 'SuiteScripts/ecdsa.p12'
});
loadedCertificate.file = fileObj;
loadedCertificate.password = '022b490ad4334c7e86a8304f937ec68f',
loadedCertificate.save();
cc.deleteCertificate({
    scriptId : 'custcertificate_testid'
});
...
// Add additional code
```

## Certificate.save()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Saves a certificate record object.
<b>Returns</b>	<a href="#">certificateControl.Certificate</a> object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/certificateControl Module</a>
<b>Parent Object</b>	<a href="#">certificateControl.Certificate</a>
<b>Sibling Object Members</b>	<a href="#">Certificate Object Members</a>
<b>Since</b>	2019.2

## 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/certificateControl Module Script Samples](#).

```
// Add additional code
...
fileObj = file.load({
    id: 'SuiteScripts/ecdsa.p12'
});
loadedCertificate.file = fileObj;
loadedCertificate.password = '022b490ad4334c7e86a8304f937ec68f',
loadedCertificate.save();
...
// Add additional code
```

## Certificate.description



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	A description of the certificate record.
<b>Type</b>	string
<b>Module</b>	<a href="#">N/certificateControl Module</a>
<b>Parent Object</b>	<a href="#">certificateControl.Certificate</a>
<b>Sibling Object Members</b>	<a href="#">Certificate Object Members</a>
<b>Since</b>	2019.2

## 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/certificateControl Module Script Samples](#).

```
// Add additional code
...
//load the certificate record object
var loadedCertificate = cc.loadCertificate({
    scriptId : 'custcertificate_testid'
});
//update the description for the certificate record
loadedCertificate.description = 'Test Certificate Description'
//save the updated certificate record
loadedCertificate.save();
...
// Add additional code
```

## Certificate.file



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The <a href="#">File Object Members</a> object of the certificate uploaded to the certificate record.
<b>Type</b>	<a href="#">File Object Members</a> object
<b>Module</b>	N/certificateControl Module
<b>Parent Object</b>	<a href="#">certificateControl.Certificate</a>
<b>Sibling Object Members</b>	<a href="#">Certificate Object Members</a>
<b>Since</b>	2019.2

## 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/certificateControl Module Script Samples](#).

```
// Add additional code
...
//load the certificate record object
var loadedCertificate = cc.loadCertificate({
   scriptId : 'custcertificate_testid'
});
//load the file from the File Cabinet
fileObj = file.load({
    id: 'SuiteScripts/ecdsa.p12'
});
//upload the file to the certificate record
loadedCertificate.file = fileObj;
//save the certificate record
loadedCertificate.save();
...
// Add additional code
```

## Certificate.name



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The name of the certificate record.
<b>Type</b>	string
<b>Module</b>	N/certificateControl Module
<b>Parent Object</b>	<a href="#">certificateControl.Certificate</a>
<b>Sibling Object Members</b>	<a href="#">Certificate Object Members</a>
<b>Since</b>	2019.2

## 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/certificateControl Module Script Samples](#).

```
// Add additional code
...
//load the certificate record object
var loadedCertificate = cc.loadCertificate({
   scriptId : 'custcertificate_testid'
});
//update the name of the certificate record
loadedCertificate.name = 'Brazil Certificate';
//save the certificate record object
loadedCertificate.save();
...
// Add additional code
```

## Certificate.monthReminder



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates the setting of the <b>Month</b> box for <b>Expiration Reminders</b> on the certificate record. This property is set to <b>true</b> if the <b>Month</b> box is checked and email reminders are sent to account administrators one month before the certificate expires.  If the <b>Copy Employees</b> box is also checked, selected employees are copied on the reminder emails.
<b>Type</b>	boolean <b>true false</b>
<b>Module</b>	<a href="#">N/certificateControl Module</a>
<b>Parent Object</b>	<a href="#">certificateControl.Certificate</a>
<b>Sibling Object Members</b>	<a href="#">Certificate Object Members</a>
<b>Since</b>	2019.2

## 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/certificateControl Module Script Samples](#).

```
// Add additional code
...
//load the certificate record object
var loadedCertificate = cc.loadCertificate({
   scriptId : 'custcertificate_testid'
});
//update the Expiration Reminder for Month to checked
loadedCertificate.monthReminder = true;
//save the certificate record object
loadedCertificate.save();
```

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

## Certificate.notifications

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal IDs of the employees copied on expiration notification email. The values for this property are found in the Copy Employees field of the Audience tab on the certificate record.  When you create or edit a certificate object with values for this property, you also check the Copy Employees box for the certificate record.
<b>Type</b>	number []
<b>Module</b>	N/certificateControl Module
<b>Parent Object</b>	certificateControl.Certificate
<b>Sibling Object Members</b>	<a href="#">Certificate Object Members</a>
<b>Since</b>	2019.2

## 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/certificateControl Module Script Samples](#).

```
// Add additional code
...
//create a variable to hold the properties of the certificate object
var options = {
    name : 'testCertp12',
    description : 'testDescription',
    scriptId : '_testidp12',
    //include the internal IDs for employees you want copied on expiration reminder email
    notifications: [168,259]
};
//create the certificate record with the options variable
var newCertificate = cc.createCertificate(options);
//save the certificate object
newCertificate.save();
...
// Add additional code
```

## Certificate.password

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The password for the digital certificate. If the certificate file is password-protected, you can store the password with the certificate record. If the certificate is not password-protected, enter an empty string.
-----------------------------	---

Type	string (write-only)
Module	N/certificateControl Module
Parent Object	certificateControl.Certificate
Sibling Object Members	Certificate Object Members
Since	2019.2

## 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/certificateControl Module Script Samples](#).

```
// Add additional code
...
//create a variable to hold the properties of the certificate object
var options = {
    name : 'testCertp12',
    description : 'testDescription',
    //for password, enter the guid associated with your digital certificate or an empty string
    password: 'yourCertPassword',
    scriptId : '_testidp12',
};
//create the certificate record with the options variable
var newCertificate = cc.createCertificate(options);
//save the certificate object
newCertificate.save();
...
// Add additional code
```

## Certificate.restrictions



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal IDs of the employees selected in the Restrict to Employees field of the certificate record. If you set this property with an employee internal ID, you check the Restrict to Employees box and select that employee.  Employees selected must also have either the Certificate Management or Certificate Access role permission in order to access the certificate. When the Restrict to Employees box is checked, only Administrators and the employees selected can access the certificate.
<b>Type</b>	number[]
<b>Module</b>	N/certificateControl Module
<b>Parent Object</b>	certificateControl.Certificate
<b>Sibling Object Members</b>	Certificate Object Members
<b>Since</b>	2019.2

## 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/certificateControl Module Script Samples](#).

```
// Add additional code
...
//load the certificate record object
var loadedCertificate = cc.loadCertificate({
   scriptId : 'custcertificate_testid'
});
//check the Restrict to Employees box and select employees with internal IDs of 189 and 250
loadedCertificate.restrictions = [189,250];
loadedCertificate.save();
...
// Add additional code
```

## Certificate.scriptId



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The ID of the certificate record. The script ID for certificate records begins with "custcertificate."
<b>Type</b>	string
<b>Module</b>	N/certificateControl Module
<b>Parent Object</b>	certificateControl.Certificate
<b>Sibling Object Members</b>	<a href="#">Certificate Object Members</a>
<b>Since</b>	2019.2

## 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/certificateControl Module Script Samples](#).

```
// Add additional code
...
//load the certificate record object
var loadedCertificate = cc.loadCertificate({
   scriptId : 'custcertificate_testid'
});
//update the text for script ID
loadedCertificate.scriptId = '_ChinaCert';
loadedCertificate.save();
...
// Add additional code
```

## Certificate.subsidiaries



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal IDs of the subsidiaries associated with the certificate record. Subsidiary selections associate a certificate to one or more subsidiaries but do not affect access.
<b>Type</b>	number[]
<b>Module</b>	N/certificateControl Module
<b>Parent Object</b>	certificateControl.Certificate
<b>Sibling Object Members</b>	Certificate Object Members
<b>Since</b>	2019.2

## 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/certificateControl Module Script Samples](#).

```
// Add additional code
...
//load the certificate record object
var loadedCertificate = cc.loadCertificate({
   scriptId : 'custcertificate_testid'
});
//set the subsidiaries to those with the internal IDs of 3 and 5
loadedCertificate.subsidiaries = [3,5];
//save the certificate record object
loadedCertificate.save();
...
// Add additional code
```

## Certificate.threeMonthsReminder



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates the setting of the <b>3 Months</b> box for <b>Expiration Reminders</b> on the certificate record. This property is set to <b>true</b> if the <b>3 Months</b> box is checked. When set to <b>true</b> , email reminders are sent to account administrators three months before the certificate expires. If the <b>Copy Employees</b> box is also checked, selected employees are copied on the reminder emails.
<b>Type</b>	boolean <b>true false</b>
<b>Module</b>	N/certificateControl Module
<b>Parent Object</b>	certificateControl.Certificate
<b>Sibling Object Members</b>	Certificate Object Members

Since	2019.2
-------	--------

## 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/certificateControl Module Script Samples](#).

```
// Add additional code
...
//load the certificate record object
var loadedCertificate = cc.loadCertificate({
   scriptId : 'custcertificate_testid'
});
//update the Expiration Reminder for 3 Months to checked
loadedCertificate.threeMonthsReminder = true;
//save the certificate record object
loadedCertificate.save();
...
// Add additional code
```

## Certificate.weekReminder

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates the setting of the <b>Week</b> box for <b>Expiration Reminders</b> on the certificate record. This property is set to <b>true</b> if the <b>Week</b> box is checked. When set to <b>true</b> , email reminders are sent to account administrators one week before the certificate expires.  If the <b>Copy Employees</b> box is also checked, selected employees are copied on the reminder emails.
<b>Type</b>	boolean <b>true false</b>
<b>Module</b>	N/certificateControl Module
<b>Parent Object</b>	certificateControl.Certificate
<b>Sibling Object Members</b>	<a href="#">Certificate Object Members</a>
<b>Since</b>	2019.2

## 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/certificateControl Module Script Samples](#).

```
// Add additional code
...
//load the certificate record object
var loadedCertificate = cc.loadCertificate({
   scriptId : 'custcertificate_testid'
});
//update the Expiration Reminder for Week to checked
```

```

loadedCertificate.weekReminder = true;
//save the certificate record object
loadedCertificate.save();
...
// Add additional code

```

## certificateControl.createCertificate(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a certificate record on the Certificates page using a file from the File Cabinet.   <b>Note:</b> Your role must have Create, Edit, or Full access to the Certificate Access permission to create certificates via SuiteScript.
<b>Returns</b>	<code>certificateControl.Certificate</code>
<b>Supported Script Types</b>	Server scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/certificateControl Module</a>
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.file</code>	file	required	A <a href="#">File Object Members</a> object. The file must already be uploaded to the File Cabinet.	2019.2
<code>options.password</code>	string	optional	If applicable, the password associated with your digital certificate.	2019.2
<code>options.scriptId</code>	string	optional	The desired script ID of the certificate record. The script ID is automatically prefixed with 'custcertificate_'	2019.2
<code>options.description</code>	string	optional	The description of the certificate record.	2019.2
<code>options.subsidiaries</code>	number[] or string[]	optional	The internal ID of subsidiaries associated with the certificate in either number or string format.	2019.2
<code>options.restrictions</code>	number[] or string[]	optional	The internal ID of employees selected in the <b>Restricted to Employees</b> field for a certificate. You can enter the internal ID in either number or string format.	2019.2
<code>options.notifications</code>	number[] or string[]	optional	The internal ID of employees selected in the <b>Copy Employees</b> field on the certificate	2019.2

Parameter	Type	Required / Optional	Description	Since
			record. You can enter the internal ID in either number or string format.	
options.name	string	required	The name of the certificate record.	2019.2
options.weekReminder	boolean true   false	optional	The setting for the <b>Expiration Reminder : Week</b> checkbox.	2019.2
options.monthReminder	boolean true   false	optional	The setting for the <b>Expiration Reminder : Month</b> checkbox.	2019.2
options.threeMonthsReminder	boolean true   false	optional	The setting for the <b>Expiration Reminder : 3 Months</b> checkbox.	2019.2

## 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/certificateControl Module Script Samples](#).

```
//Add additional code
...
var fileObj = file.load({
    id: 'SuiteScripts/dsa.p12'
});
var options = {
    file : fileObj,
    password : '022b490ad4334c7e86a8304f937ec68f',
    name : 'testCert',
    description : 'testDescription',
    scriptId : '_testid',
    subsidiaries : [1,3],
    weekReminder : false,
    monthReminder : true,
    threeMonthsReminder : false
};
var newCertificate = cc.createCertificate(options);
newCertificate.save();
...
//Add additional code
```

## certificateControl.findCertificates(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns an array of certificates available. You can use the parameters as filters for this search. If you do not use any parameters, all certificate records are returned.
<b>Returns</b>	Metadata about the certificate(s)
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Governance</b>	10 units
<b>Module</b>	N/certificateControl Module
<b>Since</b>	2019.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.subsidiary	number	optional	The internal ID of the subsidiary.	2019.1
options.type	string	optional	The certificate file type.	2019.1
options.restriction	number	optional	The internal ID of an employee selected in the <b>Restrict to Employees</b> field.	2019.2
options.notification	number	optional	The internal ID of an employee selected in the <b>Copy Employees</b> field.	2019.2
options.name	string	optional	The certificate name. You can use this filter with the <a href="#">certificateControl.Operator</a> enum.	2019.2
options.description	string	optional	The certificate description. You can use this filter with the <a href="#">certificateControl.Operator</a> enum.	2019.2

## Syntax

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

```
require(['N/certificateControl'],function(cc){
    var yodlee = cc.findCertificates({
        name: 'Yodlee',
        description: 'Yodlee certificate'
    });
})
```

## certificateControl.findUsages(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns an audit trail of how a certificate has been used. Includes operations performed with time stamps.
<b>Returns</b>	An array of operations performed.
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units

<b>Module</b>	N/certificateControl Module
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.from	date	optional	The start date for your audit trail search.	2019.2
options.to	date	optional	The end date for your audit trail search.	2019.2
options.id	string	optional	The script ID of the certificate record.	2019.2
options.operation	string	optional	The <a href="#">certificateControl.Operation</a> performed with the digital certificate.	2019.2
options.script	number	optional	The script ID of a script record that used a certificate record.	2019.2
options.deploy	number	optional	The script ID of a script deployment that used a certificate record.	2019.2
options.entity	number	optional	The internal ID of the employee who performed the operation.	2019.2

Error	Thrown If
SSS_INVALID_TYPE_ARG	A parameter provided is the wrong type.
TOO_MANY_RESULTS	There are more than 1000 results.

## Syntax

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

```
//Add additional code
...
var usages = cc.findUsages({
    id: 'custcertificate_china',
    operation: cc.Operation.POST
});
...
//Add additional code
```

## certificateControl.deleteCertificate(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Deletes a certificate record that has been uploaded to the Certificates list in the UI or created using <a href="#">certificateControl.createCertificate(options)</a> .
---------------------------	---

	<p><b>Note:</b> Your role must have either Edit or Full access to the Certificate Access permission to delete certificate records via SuiteScript. History of the certificate is not deleted.</p>
Returns	The script ID of the deleted certificate.
Supported Script Types	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Governance	10 units
Module	N/certificateControl Module
Since	2019.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.								
<table border="1"> <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 or internal ID for the certificate you want to delete. You can view the ID of a certificate from the Digital Certificates list at Setup &gt; Company &gt; Certificates.</td></tr> </tbody> </table>	Parameter	Type	Required / Optional	Description	options.scriptId	string	required	The script ID or internal ID for the certificate you want to delete. You can view the ID of a certificate from the Digital Certificates list at Setup > Company > Certificates.
Parameter	Type	Required / Optional	Description					
options.scriptId	string	required	The script ID or internal ID for the certificate you want to delete. You can view the ID of a certificate from the Digital Certificates list at Setup > Company > Certificates.					

## Syntax

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

```
// Add additional code
...
define(['N/certificateControl'],function(cc){
    var usages = cc.deleteCertificate({
       scriptId: 'custcertificate_china'
    });
})
...
// Add additional code
```

## certificateControl.loadCertificate(options)

<b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.				
<table border="1"> <thead> <tr> <th>Method Description</th><th>Loads a certificate record that has been uploaded to the Certificates list in the UI or created using <a href="#">certificateControl.createCertificate(options)</a>.</th></tr> </thead> <tbody> <tr> <td>Returns</td><td><a href="#">certificateControl.Certificate</a></td></tr> </tbody> </table>	Method Description	Loads a certificate record that has been uploaded to the Certificates list in the UI or created using <a href="#">certificateControl.createCertificate(options)</a> .	Returns	<a href="#">certificateControl.Certificate</a>
Method Description	Loads a certificate record that has been uploaded to the Certificates list in the UI or created using <a href="#">certificateControl.createCertificate(options)</a> .			
Returns	<a href="#">certificateControl.Certificate</a>			

<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/certificateControl Module</a>
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.scriptId	string	required	The script ID or internal ID for the certificate you want to load. You can view the ID of a certificate from the <b>Digital Certificates</b> list at Setup > Company > Certificates.

## Syntax



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

```
// Add additional code
...
//load the certificate record object
var loadedCertificate = cc.loadCertificate({
   scriptId : 'custcertificate_testid'
});
//load a digital certificate from the File Cabinet
fileObj = file.load({
    id: 'SuiteScripts/ecdsa.p12'
});
//upload the file to the certificate record
loadedCertificate.file = fileObj;
//update the password to match the guid password for the certificate
loadedCertificate.password = 'certPass';
//save the certificate
loadedCertificate.save();
...
// Add additional code
```

## certificateControl.Operation



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the values for the operation parameter of .
-------------------------	--

	<p><b>Note:</b> 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>
<b>Module</b>	<a href="#">N/certificateControl Module</a>
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Since</b>	2019.2

## Values

- CONNECT
- DELETE
- FIND
- GET
- HEAD
- POST
- PUT
- SIGN\_STRING
- SIGN\_XML
- VERIFY\_STRING
- VERIFY\_XML

## Syntax



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

```
//Add additional code
...
var usages = cc.findUsages({
    id: 'custcertificate_china',
    operation: cc.Operation.POST
});
...
//Add additional code
```

## certificateControl.Operator



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Search operators to use with the <code>name</code> and <code>description</code> parameters of the <code>certificateControl.findCertificates(options)</code> .
-------------------------	---

	<p><b>Note:</b> 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>
<b>Module</b>	N/certificateControl Module
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Since</b>	2019.2

## Values

- CONTAINS
- ENDS\_WITH
- EQUALS
- STARTS\_WITH

## certificateControl.Type

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	The certificate file type. PFX, PEM, and P12 are supported.
	<p><b>Note:</b> 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>
<b>Module</b>	N/certificateControl Module
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Since</b>	2019.1

## Values

- PFX
- P12
- PEM

## Syntax

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

```
// Add additional code
```

```

...
var specificType = certificateControl.findCertificates({
    type: certificateControl.Type.PFX
});
...
// Add additional code

```

## N/commerce Modules

**i Applies to:** Commerce Web Stores

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

Modules in the N/commerce namespace have been designed for use with SSP applications written in SuiteScript 2.x. Their primary objective is to provide web developers a homogeneous development experience when working with Commerce and SuiteScript APIs. Note that N/commerce itself is not a module.

Developers can use modules in the N/commerce namespace to access different assets in the web store context, such as items and shopping cart. The modules within the N/commerce namespace are supported by the latest version of SuiteCommerce and by SuiteCommerce Advanced 2019.2 onwards.

Before you can load N/commerce modules, you must have an active shopping session.

The modules available within the N/commerce namespace are:

Module	Description
N/commerce/recordView Module	Load the N/commerce/recordView module when you want to provide fast, cached, and public access to the item fields and website settings.

**i Note:** Commerce modules are not yet available for all web store interactions. To fully customize your web store on the SuiteCloud platform, you can use Commerce APIs with SSP applications written in SuiteScript 1.0. However, Commerce APIs are not available for use with SSP applications written in SuiteScript 2.x. For more information about the Commerce APIs, see the help topic [Commerce API](#).

## N/commerce/recordView Module

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

Load the N/commerce/recordView module when you want to provide fast, cached, and public access to the item fields and website settings.

- [N/commerce/recordView Module Members](#)
- [N/commerce/recordView Script Sample](#)

## N/commerce/recordView Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">recordView.viewItems (options)</a>	Object   array	Client and server-side scripts	Retrieves one or more Items with requested

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
		Returns one or more Items with requested items fields as an object with <b>field:value</b> pairs.  See the <a href="#">Returns</a> section.		items fields from an Item Record.
	recordView.viewWebsite(options)	Object  Returns website and website fields as an object with <b>field:value</b> pairs.  See the <a href="#">Returns</a> section.	Client and server-side scripts	Retrieves the website details with requested website fields.

## Ncommerce/recordView Script Sample

The following example retrieves some details of the website and some item data for the specified items.

**i Note:** This sample script uses the define function, which is used in your entry point script (the script you attach to a script record). Keep in mind that you must use the require function instead of define if you want to copy it into the debugger and test it.

For help with writing scripts in SuiteScript 2.x, see [SuiteScript 2.0 Hello World](#) and [SuiteScript 2.0 Entry Point Script Creation and Deployment](#).

```
/*
 * @NApiVersion 2.x
 */
define(['Ncommerce/recordView'],
    function (recordView) {
        function service(context) {
            var result = {};

            try {
                result.website= recordView.viewWebsite({
                    id: 2,
                    fields: ["internalid","shiptocountries","websitehomepage"]
                });
            }
            catch (e) {
                result.websiteError = e.name + " : " + e.message;
            }

            var options = {
                "ids": [382,388],
                "fields": ["displayname","welcomedescription"]
            };
            try {
                result.items= recordView.viewItems(options);
            }
            catch (e) {
                result.itemsError = e.name + " : " + e.message;
            }
        }
    }
);
```

```

        return context.response.write(
            JSON.stringify(result)
        );
    }

    return {
        service: service
    };
}
);

```

## recordView.viewItems(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Retrieves one or more items with requested item fields from an Item Record.
<b>Returns</b>	See the <a href="#">Returns</a> section.
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">Ncommerce/recordView Module</a>
<b>Sibling Module Members</b>	<a href="#">Ncommerce/recordView Module Members</a>
<b>Since</b>	2019.1

## Parameters

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

Property	Type	Required / Optional	Description
options.ids	number[]	required	IDs of the item you want to view
options.fields	string   string[]	required	Item fields you want to retrieve for the items
options.fieldOptions	Array of name, value pairs. Type depends upon parameter:  includeVat - string   boolean[]	optional	Options that affect related fields.  Supported field options:  includeVat - this affects onlinecustomerprice_detail field. Default value is false.

## Errors

Error Code	Thrown If
SSS_INVALID_TYPE_ARG	Parameter is invalid
FIELD_1_CANNOT_BE_EMPTY	Required parameter is missing or empty

## Syntax

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

```
// Add additional code here
...
try {
    result.viewItems = recordView.viewItems({
        ids: [408],
        fields: ["itemtype", "isavailable"]
        fieldOptions: [{"includeVat":true}]
    });
}
catch (e) {
    result.error = e.name + ": " + e.message;
}
...
...
```

## Returns

Returns a flat JSON structure with **field:value** pairs.

```
[{
    "internalid": {
        value : 523
    },
    "name": {
        value : "test"
    },
    "dropdownListField": {
        value : {
            "id": 1,
            "label": "Option 1"
        }
    },
    "checkbox1Field": {
        value : true
    },
    "dateField1Field": {
        value : "2012-04-23T18:25:43.511Z"
    }
}, ... ]
```

## recordView.viewWebsite(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Retrieves the website details with requested website fields.
<b>Returns</b>	See the <a href="#">Returns</a> section.

<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	Ncommerce/recordView Module
<b>Sibling Module Members</b>	Ncommerce/recordView Module Members
<b>Since</b>	2019.1

## Parameters

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

Property	Type	Required / Optional	Description
options.id	number	required	ID of the website
options.fields	string   string[]	required	Website fields you want to retrieve
options.fieldOptions	Array of name, value pairs. Type depends upon parameter.	optional	Options that affect all related fields that are retrieved.  Supported field options: None

## Errors

Error Code	Thrown If
SSS_INVALID_TYPE_ARG	Parameter is invalid
FIELD_1_CANNOT_BE_EMPTY	Required parameter is empty

## Syntax



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

```
// Add additional code here
...
try {
    result.viewItems = recordView.viewWebsite({
        id: 2,
        fields: ["shiptocountries"]
    });
}
catch (e) {
    result.error = e.name + ": " + e.message;
}
...
```

## Returns

Returns a flat JSON structure with **field:value** pairs.

```
[{
    "internalid": {
        value : 523
    },
    "name": {
        value : "test"
    },
    "dropdownListField": {
        value : {
            "id": 1,
            "label":"Option 1"
        }
    },
    "checkbox1Field": {
        value : true
    },
    "dateField1Field": {
        value : "2012-04-23T18:25:43.511Z"
    }
}, ... ]
```

## N/config Module



**Note:** The content in this help topic pertains to SuiteScript 2.0.

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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">config.load(options)</a>	<a href="#">record.Record</a>	Server-side scripts	Loads a <a href="#">record.Record</a> object that encapsulates the specified configuration page.
Enum	<a href="#">config.Type</a>	enum	Server-side scripts	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.

## N/config Module Script Sample

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

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

**i 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.

```
/*
 * @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'
            });
            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)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">Preference Names and IDs</a> .
---------------------------	---

	You can use the following Record Object Members to get and set preference names and IDs: <ul style="list-style-type: none"><li>▪ <a href="#">Record.getField(options)</a></li><li>▪ <a href="#">Record.getFields()</a></li><li>▪ <a href="#">Record.getText(options)</a></li><li>▪ <a href="#">Record.getValue(options)</a></li><li>▪ <a href="#">Record.setText(options)</a></li><li>▪ <a href="#">Record.setValue(options)</a></li></ul>
<b>Returns</b>	<a href="#">record.Record</a>
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 usage units
<b>Module</b>	<a href="#">N/config Module</a>
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
<code>options.type</code>	enum	required	The NetSuite configuration page you want to access. Use the <a href="#">config.Type</a> enum to set the value.	2015.2
<code>options.isDynamic</code>	boolean <code>true</code>   <code>false</code>	optional	<p>Determines whether the record is loaded in dynamic mode.</p> <ul style="list-style-type: none"> <li>▪ If set to <code>true</code>, the record is loaded in dynamic mode.</li> <li>▪ If set to <code>false</code>, the record is loaded in standard mode.</li> </ul> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a>.</p>	2015.2

Error Code	Message	Thrown If
<code>INVALID_RCRD_TYPE</code>	The record type {type} is invalid.	The <code>type</code> argument is invalid or missing.

## Syntax

 <b>Important:</b>	The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/config Module Script Sample</a> .
---	---

```
//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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for supported configuration pages.
	 <b>Note:</b> 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.
<b>Module</b>	N/config Module
<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Since</b>	2015.2

## Values

<b>Value</b>	<b>Configuration Page</b>
USER_PREFERENCES	Set Preferences page (Home > Set Preferences) For more information about the fields on the page, see the help topic <a href="#">User Preferences</a> .
COMPANY_INFORMATION	Company Information page (Setup > Company > Company Information) For more information about the fields on the page, see the help topic <a href="#">Company Information</a> .
COMPANY_PREFERENCES	General Preferences page (Setup > Company > General Preferences) For more information about the fields on the page, see the help topic <a href="#">General Preferences</a> .
ACCOUNTING_PREFERENCES	Accounting Preferences page (Setup > Accounting > Accounting Preferences) For more information about the fields on the page, see the help topic <a href="#">Accounting Preferences</a> .
ACCOUNTING_PERIODS	Accounting Periods page (Setup > Accounting > Manage Accounting Periods) For more information about the fields on the page, see the help topic <a href="#">Accounting Periods</a> .

Value	Configuration Page
TAX_PERIODS	Tax Periods page (Setup > Accounting > Manage Tax Periods) For more information about the fields on the page, see the help topic <a href="#">Tax Periods</a> .
FEATURES	Enable Features page (Setup > Company > Enable Features) For more information about feature names and IDs, see the help topic <a href="#">Feature Names and IDs</a> .
TIME_POST	For additional information, see the help topic <a href="#">Posting Time Transactions</a> .
TIME_VOID	For additional information, see the help topic <a href="#">Posting Time Transactions</a> .

## Syntax



**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](#).

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

## N/crypto Module



**Note:** The content in this help topic pertains to SuiteScript 2.0.

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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">crypto.Cipher</a>	Object	Server-side scripts	Encapsulates a cipher.
	<a href="#">crypto.CipherPayload</a>	Object	Server-side scripts	Encapsulates a cipher payload.
	<a href="#">crypto.Decipher</a>	Object	Server-side scripts	Encapsulates a decipher.
	<a href="#">crypto.Hash</a>	Object	Server-side scripts	Encapsulates a hash.
	<a href="#">crypto.Hmac</a>	Object	Server-side scripts	Encapsulates an hmac.
	<a href="#">crypto.SecretKey</a>	Object	Server-side scripts	Encapsulates a secret key handle.
Method	<a href="#">crypto.createCipher(options)</a>	Object	Server-side scripts	Creates and returns a new <a href="#">crypto.Cipher</a> Object.
	<a href="#">crypto.createDecipher(options)</a>	Object	Server-side scripts	Creates and returns a new <a href="#">crypto.Decipher</a> object.
	<a href="#">crypto.createHash(options)</a>	Object	Server-side scripts	Creates and returns a new <a href="#">crypto.Hash</a> Object.
	<a href="#">crypto.createHmac(options)</a>	Object	Server-side scripts	Creates and returns a new <a href="#">crypto.Hmac</a> Object.
	<a href="#">crypto.createSecretKey(options)</a>	Object	Server-side scripts	Creates and returns a new <a href="#">crypto.SecretKey</a> Object.
Enum	<a href="#">crypto.EncryptionAlg</a>	string (read-only)	Server-side scripts	Holds the string values for supported encryption algorithms. Sets the <code>options.algorithm</code> parameter for <a href="#">crypto.createCipher(options)</a> .
	<a href="#">crypto.HashAlg</a>	string (read-only)	Server-side scripts	Holds the string values for supported hashing algorithms. Sets the value of the <code>options.algorithm</code> parameter for <a href="#">crypto.createHash(options)</a> and <a href="#">crypto.createHmac(options)</a> .
	<a href="#">crypto.Padding</a>	string (read-only)	Server-side scripts	Holds the string values for supported cipher padding. Sets the <code>options.padding</code> parameter for <a href="#">crypto.createCipher(options)</a> and <a href="#">crypto.createDecipher(options)</a> .

## Cipher Object Members

The following members are called on [crypto.Cipher](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">Cipher.update(options)</a>	Object	Server-side scripts	Updates the clear data with the specified encoding
	<a href="#">Cipher.final(options)</a>	Object	Server-side scripts	Returns the cipher data.

## CipherPayload Object Members

The following members are called on [crypto.CipherPayload](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	CipherPayload.ciphertext	string	Server-side scripts	The result of the ciphering process.
	CipherPayload.iv	number	Server-side scripts	An initialization vector

## Decipher Object Members

The following members are called on [crypto.Decipher](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Decipher.final(options)	string	Server-side scripts	Returns the clear data.
	Decipher.update(options)	void	Server-side scripts	Updates cipher data with the specified encoding.

## Hash Object Members

The following members are called on [crypto.Hash](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Hash.digest(options)	string	Server-side scripts	Calculates the digest of the data to be hashed.
	Hash.update(options)	void	Server-side scripts	Updates the clear data with the encoding specified.

## Hmac Object Members

The following members are called on [crypto.Hmac](#).

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

## SecretKey Object Members

The following members are called on [crypto.SecretKey](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	Secretkey.guid	string	Server-side scripts	The GUID associated with the secret key.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	<a href="#">SecretKey.encoding</a>	string	Server-side scripts	The encoding used for the clear text value of the secret key.

## N/crypto Module Script Samples

The following script samples demonstrate how to use the features of the N/crypto module.

### Sample 1: Generate a secure key

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample demonstrates the APIs needed to generate a secure key using the SHA512 hashing algorithm. The GUID in this sample is a placeholder. You must replace it with a valid value from your NetSuite account. 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 sample that shows creating a form field that generates a GUID.

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

require(['N/crypto', 'N/encode', 'N/runtime'], function(crypto, encode, runtime) {
    function createSecureKeyWithHash() {
        var inputString = 'YWJjZGVmZwo=';
        var myGuid = '{284CFB2D225B1D76FB94D150207E49DF}';

        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();
});
```

## Sample 2: Create a Suitelet to work with keys

**Note:** This script sample uses the `define` function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the `require` function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample creates a simple Suitelet that requests user credentials, creates a secret key, and encodes a sample string. 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
            });

            try {
                var hmacSha512 = crypto.createHmac({
                    algorithm: 'SHA512',
                    key: sKey
                });
                hmacSha512.update({
                    input: inputString,
                    inputEncoding: encode.Encoding.BASE_64
                });
            }
        }
    }
});
```

```

    });
    var digestSha512 = hmacSha512.digest({
        outputEncoding: encode.Encoding.HEX
    });
} catch (e) {
    log.error({
        title: 'Failed to hash input',
        details: e
    });
}

form.addField({
    id: 'result',
    label: 'Your digested hash value',
    type: 'textarea'
}).defaultValue = digestSha512;

option.response.writePage(form);
}
}

return {
    onRequest: onRequest
};
});

```

## crypto.Cipher



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a cipher. For a complete list of this object's methods and properties, see <a href="#">Cipher Object Members</a> .
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	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 Sample](#).

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

```
});  
...  
//Add additional code
```

## Cipher.final(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to return the cipher data.  Sets the output encoding for the <a href="#">crypto.CipherPayload</a> object.
<b>Returns</b>	A <a href="#">crypto.CipherPayload</a> Object
<b>Supported Script Types</b>	Server-side scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
options.outputEncoding	enum	optional	The output encoding for a <a href="#">crypto.CipherPayload</a> object.  The default value is HEX.  Use the <a href="#">encode.Encoding</a> enum to set the 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/crypto Module Script Sample](#).

```
//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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

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

### Parameters



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

Parameter	Type	Required / Optional	Description
options.input	string	required	The clear data to be updated.
options.inputEncoding	enum	optional	The input encoding. Use the <a href="#">encode.Encoding</a> enum to set the value. The default value is <b>UTF_8</b> .

### Syntax



**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 Sample](#).

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

## crypto.CipherPayload



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a cipher payload.
---------------------------	--------------------------------

	For a complete list of this object's methods and properties, see <a href="#">CipherPayload Object Members</a> .
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	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 Sample](#).

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

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

## CipherPayload.ciphertext

**ℹ Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The result of the ciphering process. For example, to take the cipher payload and send it to another system.
<b>Type</b>	string
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	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 Sample](#).

```
//Add additional code
...
log.debug({
```

```

        title: "Ciphertext: ",
        details: cipherPayload.ciphertext
    });
    ...
//Add additional code

```

## CipherPayload.iv



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Initialization vector for the cipher payload. You can pass in the iv value to <a href="#">crypto.createDecipher(options)</a>
<b>Type</b>	string
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	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 Sample](#).

```

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

```

## crypto.Decipher



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a decipher. This object has methods that decrypt. For a complete list of this object's methods and properties, see <a href="#">Decipher Object Members</a> .
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/crypto Module</a>

<b>Since</b>	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 Sample.

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

## Decipher.final(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to return the clear data.
<b>Returns</b>	string
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/crypto Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.outputEncoding	string	optional	Specifies the encoding for the output Set the value using the <a href="#">encode.Encoding</a> enum. The default value is <b>UTF_8</b> .

## Syntax

**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 Sample.

```
//Add additional code
```

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

## Decipher.update(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to update cipher data with the specified encoding.
<b>Returns</b>	Void
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description
options.input	string	required	The data to update
options.inputEncoding	string	optional	Specifies the encoding of the input data Set the value using the <a href="#">encode.Encoding</a> enum. The default value is 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/crypto Module Script Sample](#).

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

# crypto.Hash

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a hash. For a complete list of this object's methods and properties, see <a href="#">Hash Object Members</a> .
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	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 Sample](#).

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

## Hash.digest(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Calculates the digest of the data to be hashed.
<b>Returns</b>	A hash value as a string
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.outputEncoding	string	optional	The output encoding. Set using the <a href="#">encode.Encoding</a> enum.

Parameter	Type	Required / Optional	Description
			The default value is 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/Module Script Sample.

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

## Hash.update(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

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

## Parameters

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

Parameter	Type	Required / Optional	Description
options.input	string	required	The data to be updated.
options.inputEncoding	string	optional	The input encoding. Set using the <a href="#">encode.Encoding</a> enum. The default value is <a href="#">UTF_8</a> .

## Syntax

**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 Sample.

```
//Add additional code
```

```

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

```

## crypto.Hmac



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates an hmac. For a complete list of this object's methods and properties, see <a href="#">Hmac Object Members</a> .
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	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 Sample](#).

```

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

```

## Hmac.digest(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the computed digest.
<b>Returns</b>	An hmac value as a string
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None

<b>Module</b>	N/crypto Module
<b>Since</b>	2015.2

## Parameters

<b>i Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description
options.outputEncoding	string	optional	Specifies the encoding of the output string. Set using the <a href="#">encode.Encoding</a> enum. The default value is HEX.

## Syntax

<b>⚠ Important:</b> 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 Sample.
--

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

## Hmac.update(options)

<b>i Note:</b> The content in this help topic pertains to SuiteScript 2.0.
--

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

## Parameters

<b>i Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description
options.input	string	required	The hmac data to be updated.

Parameter	Type	Required / Optional	Description
options.inputEncoding	enum	optional	The input encoding. Set using the <a href="#">encode.Encoding</a> enum. The default value is <code>UTF_8</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/crypto Module Script Sample.

```
//Add additional code
...
hmacSHA512.update({
    input: inputString,
    inputEncoding: encode.Encoding.BASE_64
});
...
//Add additional code
```

## crypto.SecretKey

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	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 <a href="#">SecretKey Object Members</a> .
<b>Supported Script Types</b>	Server-side scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	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 Sample.

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

## SecretKey.encoding



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The encoding used for the clear text value of the secret key.
<b>Type</b>	string
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	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 Sample](#).

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

## Secretkey.guid



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The GUID associated with the secret key.
<b>Type</b>	string
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	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 Sample](#).

```
//Add additional code
...
log.debug({
    title: "Secret Key GUID: ",
```

```

    details: sKey.guid
});
...
//Add additional code

```

## crypto.createCipher(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to create and return a <a href="#">crypto.EncryptionAlg</a> object.
	 <b>Note:</b> The blockCipherMode is automatically set to CBC.
<b>Returns</b>	A <a href="#">crypto.EncryptionAlg</a> object
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.algorithm	string	required	The hash algorithm. Set the value using the <a href="#">crypto.EncryptionAlg</a> enum.	2015.2
options.key	object	required	The <a href="#">crypto.SecretKey</a> object.   <b>Note:</b> When using the <a href="#">crypto.SecretKey</a> 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.	2015.2
options.padding	string	optional	The padding for the cipher text. Set the value using the <a href="#">crypto.Padding</a> enum. By default, the value is set to PKCS5Padding.	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 Sample](#).

```
//Add additional code
```

```
...
var cipher = crypto.createCipher({
    algorithm: crypto.EncryptionAlg.AES,
    key: sKey,
    padding: crypto.Padding.PKCS5Padding
});
...
//Add additional code
```

## crypto.createDecipher(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to create a <a href="#">crypto.Decipher</a> object.   <b>Note:</b> The blockCipherMode is automatically set to CBC.
<b>Returns</b>	A <a href="#">crypto.Decipher</a> object.
<b>Supported Script Types</b>	Server-side scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.algorithm	string	required	The hash algorithm. Set by the <a href="#">crypto.EncryptionAlg</a> enum.	2015.2
options.key	object	required	The <a href="#">crypto.SecretKey</a> object used for encryption.   <b>Note:</b> When using the <a href="#">crypto.SecretKey</a> 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.	2015.2
options.padding	object	optional	The padding for the cipher. Set the value using the <a href="#">crypto.Padding</a> enum.	2015.2
options.iv	string	required	The initialization vector that was used for encryption.	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 Sample.

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

## crypto.createHash(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to create a <a href="#">crypto.Hash</a> object.
<b>Returns</b>	The <a href="#">crypto.Hash</a> object created using this method.
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.algorithm	string	required	■ The hash algorithm. Set using the <a href="#">crypto.HashAlg</a> enum.	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 Sample.

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

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

## crypto.createHmac(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to create a <a href="#">crypto.Hmac</a> object.
<b>Returns</b>	A <a href="#">crypto.Hmac</a> object.
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.algorithm	string	required	The hash algorithm. Use the <a href="#">crypto.HashAlg</a> enum to set this value.	2015.2
options.key	object	required	The <a href="#">crypto.SecretKey</a> object.	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 Sample](#).

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

## crypto.createSecretKey(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to create a new <a href="#">crypto.SecretKey</a> object. This method can take a GUID. Use <a href="#">Form.addCredentialField(options)</a> to generate a value.
---------------------------	--

	<p><b>Note:</b> When using the <code>crypto.SecretKey</code> 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.</p>
<b>Returns</b>	A <code>crypto.SecretKey</code> object
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Since</b>	2015.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

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

## Syntax

<b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/crypto Module Script Sample</a> .
---

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

## crypto.EncryptionAlg

<b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.
<b>Enum Description</b> Holds the string values for supported encryption algorithms. Sets the <code>options.algorithm</code> parameter for <code>crypto.createCipher(options)</code> .

	<p><b>Note:</b> 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>
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Since</b>	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 Sample](#).

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

## crypto.HashAlg

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the string values for supported hashing algorithms. Sets the value of the <code>options.algorithm</code> parameter for <code>crypto.createHash(options)</code> and <code>crypto.createHmac(options)</code> .
	<p><b>Note:</b> 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>
<b>Module</b>	<a href="#">N/crypto Module</a>
<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Since</b>	2015.2
--------------	--------

## 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 Sample.

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

## crypto.Padding



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	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> .
	<p> <b>Note:</b> 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>
<b>Module</b>	N/crypto Module
<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Since</b>	2015.2

## Values

- NoPadding

- PKCS5Padding

## Syntax



**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 Sample.

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

# N/crypto/certificate Module



**Note:** The content in this help topic pertains to SuiteScript 2.0.

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](#)
- [SignedXml Object Members](#)
- [Verifier Object Members](#)
- [N/crypto/certificate Module Script Samples](#)

## N/crypto/certificate Module Members

Member Type	Name	Return Type/ Value Type	Supported Script Types	Description
Object	<a href="#">certificate.SignedXml</a>	Object	Server scripts	Object for an XML string that has been digitally signed. Use <a href="#">certificate.signXml(options)</a> to create this object.
	<a href="#">certificate.Signer</a>	Object	Server scripts	Object for creating signatures for plain strings. Use <a href="#">certificate.createSigner(options)</a> to create this object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	<a href="#">certificate.Verifier</a>	Object	Server scripts	Object for verifying plain string signatures. Use <a href="#">certificate.createVerifier(options)</a> to create this object.
Method	<a href="#">certificate.verifyXmlSignature(options)</a>	void	Server scripts	Verifies the signature in the <a href="#">SignedXml.asFile()</a> file.
	<a href="#">certificate.createSigner(options)</a>	<a href="#">certificate.Signer</a>	Server scripts	Creates a signer object for signing plain strings.
	<a href="#">certificate.createVerifier(options)</a>	<a href="#">certificate.Verifier</a>	Server scripts	Creates a verifier object for verifying signatures of plain strings.
	<a href="#">certificate.signXml(options)</a>	<a href="#">certificate.SignedXml</a>	Server scripts	Signs the input XML string using the Certificate ID. Returns the <a href="#">SignedXml</a> as a string.
Enum	<a href="#">certificate.HashAlg</a>	enum	Server scripts	Holds the values for hash algorithms. SHA1, SHA256, SHA384, or SHA512 are supported digest methods and values for this enum.

## Signer Object Members

The following members are called on the [certificate.Signer](#) object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">Signer.sign(options)</a>	void	Server scripts	Signs the string and returns the signature.
	<a href="#">Signer.update(options)</a>	void	Server scripts	Updates the input string to be signed. The string can be encoded.

## Verifier Object Members

The following members are called on the [certificate.Verifier](#) object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">Verifier.update(options)</a>	void	Server scripts	Updates the string to be verified against specified certificate.
	<a href="#">Verifier.verify(options)</a>	void	Server scripts	Verifies the string against provided signature using specified certificate.

## SignedXml Object Members

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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<code>SignedXml.asFile()</code>	<code>file.File</code>	Server scripts	Returns the signed XML as a file object.
	<code>SignedXmlasString()</code>	<code>string</code>	Server scripts	Returns the signed XML as a string.
	<code>SignedXml.asXml()</code>	<code>xml.Document</code>	Server scripts	Returns the signed XML as an XML document. You can use the <a href="#">N/xml Module</a> with this document to access elements and attributes in the XML.

## N/crypto/certificate Module Script Samples

The following script samples demonstrate how to use the features of the N/crypto/certificate module.

### Sample 1: Load and Sign XML File

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample loads an XML file from the File Cabinet and signs it using the digital certificate with internal ID 'custcertificate1'.

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

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

## Sample 2: Create Signer and Verifier Objects

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

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

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

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

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	A signed XML string. This object is returned by the <code>certificate.signXml(options)</code> method.
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/crypto/certificate Module</a>
<b>Methods and Properties</b>	<a href="#">SignedXml Object Members</a>
<b>Since</b>	2019.1

## Syntax

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

```
//Add additional code
```

```

...
var signedXml = cert.signXML({
    algorithm: 'SHA1',
    certId: 'custcertificate1',
    rootTag: 'infNFe',
    xmlString: infNFe.getContents()
});
certificate.verifyXMLSignature({
    signedXml:signedXml,
    rootTag: 'infNFe'
});

...
//Add additional code

```

## SignedXml.asFile()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the signed XML as a file.
<b>Returns</b>	<a href="#">file.File</a>
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/crypto/certificate Module</a>
<b>Parent Object</b>	<a href="#">certificate.SignedXml</a>
<b>Sibling Object Members</b>	SignedXml Object Members
<b>Since</b>	2019.2

## SignedXmlAsString()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the signed XML as a string.
<b>Returns</b>	string
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/crypto/certificate Module</a>
<b>Parent Object</b>	<a href="#">certificate.SignedXml</a>
<b>Sibling Object Members</b>	SignedXml Object Members

<b>Since</b>	2019.1
--------------	--------

## SignedXml.asXml()

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Returns the signed XML as an XML document.
<b>Returns</b>	<code>xml.Document</code>
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/crypto/certificate Module</a>
<b>Parent Object</b>	<a href="#">certificate.SignedXml</a>
<b>Sibling Object Members</b>	<a href="#">SignedXml Object Members</a>
<b>Since</b>	2019.2

## certificate.Signer

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Object Description</b>	Object used for signing plain strings. This object is returned by the <a href="#">certificate.createSigner(options)</a> method.
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/crypto/certificate Module</a>
<b>Methods and Properties</b>	<a href="#">Signer Object Members</a>
<b>Since</b>	2019.1

## Signer.update(options)

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Updates the input string to be signed. The string can be encoded.
<b>Returns</b>	<code>void</code>
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None

<b>Module</b>	N/crypto/certificate Module
<b>Parent Object</b>	certificate.Signer
<b>Sibling Object Members</b>	Signer Object Members
<b>Since</b>	2019.1

## Parameters

<p><b>Note:</b> The options parameter is a JavaScript object.</p>			
Parameter	Type	Required / Optional	Description
options.input	string	required	The string to update.

## Errors

Error Code	Thrown If
SSS_UNSUPPORTED_ENCODING	The value for is invalid. This must be a text value, such as UTF-8, ISO_8859_1, ASCII. Values from encode.Encoding (N/encode module) are not valid.

## Syntax

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

```
//Add additional code
...
var signer = certificate.createSigner({
    certId: 'custcertificate1',
    algorithm: 'SHA1'
});
signer.update("test");
var result = signer.sign();
...
//Add additional code
```

## Signer.sign(options)

<p><b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.</p>	
<p><b>Method Description</b></p>	<p>Signs the string and returns the signature.</p>

<b>Returns</b>	string
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/crypto/certificate Module</a>
<b>Parent Object</b>	<a href="#">certificate.Signer</a>
<b>Sibling Object Members</b>	<a href="#">Signer Object Members</a>
<b>Since</b>	2019.1

## Parameters

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

Parameter	Type	Required / Optional	Description
options.outputEncoding	string	optional	Encoding of the signed string in Base64 format.

## Syntax

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

```
//Add additional code
...
var signer = certificate.createSigner({
    certId: 'custcertificate1',
    algorithm: 'SHA1'
});
signer.update("test");
var result = signer.sign();
...
//Add additional code
```

## certificate.Verifier

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Object for verifying plain string signatures. This object is returned by the <a href="#">certificate.createVerifier(options)</a> method.
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/crypto/certificate Module</a>

<b>Methods and Properties</b>	Verifier Object Members
<b>Since</b>	2019.1

## Syntax

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

```
//Add additional code
...
var verifier = certificate.createVerifier({
    certId: 'custcertificate1',
    algorithm: 'SHA1'
});
verifier.update('test');
verifier.verify(result);
...
//Add additional code
```

## Verifier.update(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Updates the string to be verified against a specified certificate.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/crypto/certificate Module</a>
<b>Parent Object</b>	Parameters
<b>Sibling Object Members</b>	Verifier Object Members
<b>Since</b>	2019.1

## Parameters

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

Parameter	Type	Required / Optional	Description
options.input	string	required	The string to verify.
options.inputEncoding	string	optional	Encoding of the string to verify. The default value is UTF-8.

## Verifier.verify(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Verifies a string against a provided signature using a specified certificate. You can create a verifier object using the <a href="#">certificate.createVerifier(options)</a> method.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/crypto/certificate Module</a>
<b>Parent Object</b>	Parameters
<b>Sibling Object Members</b>	Verifier Object Members
<b>Since</b>	2019.1

## Parameters

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

Parameter	Type	Required / Optional	Description
options.signature	string	required	The signature to be verified.
options.signatureEncoding	string	optional	The signature's encoding in Base64 format.

## Errors

Error Code	Thrown If
INVALID_SIGNATURE	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.

## certificate.createSigner(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates the signer object for signing plain strings.
<b>Returns</b>	<a href="#">certificate.Signer</a>
<b>Supported Script Types</b>	Server scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units

<b>Module</b>	N/crypto/certificate Module
<b>Since</b>	2019.1

## Parameters

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

Parameter	Type	Required / Optional	Description
options.certId	string	required	The script ID of the digital certificate.
options.algorithm	string	required	The hash algorithm.

## certificate.createVerifier(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates the verifier object for verifying signatures of plain strings.
<b>Returns</b>	A <a href="#">Parameters</a> object
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	N/crypto/certificate Module
<b>Since</b>	2019.1

## Parameters

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

Parameter	Type	Required / Optional	Description
options.certId	string	required	The script ID of the digital certificate.
options.algorithm	string	required	Hash algorithm

## certificate.verifyXmlSignature(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Verifies the signature in the <code>signedXml</code> object or string.
---------------------------	--

<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/crypto/certificate Module</a>
<b>Since</b>	2019.1

## Parameters

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

Parameter	Type	Required / Optional	Description
options.signedXml	string	required	Signed XML
options.rootTag	string	required	Signed root XML tag.
options.certId	string	optional	The script ID for the digital certificate.

## certificate.signXml(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Signs the inputXml string using the certId.
<b>Returns</b>	<a href="#">certificate.SignedXml</a>
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/crypto/certificate Module</a>
<b>Since</b>	2019.1

## Parameters

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

Parameter	Type	Required / Optional	Description
options.xmlString	string	required	Input XML string
options.certId	string	required	Certificate ID
options.algorithm	string	required	Hash algorithm

Parameter	Type	Required / Optional	Description
options.rootTag	string	required	Root tag of XML section to sign
options.insertionTag	string	optional	Tag where to insert the signature

## certificate.HashAlg

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	The hash algorithm.  Supported digest methods are SHA1, SHA256, SHA384, and SHA512 for RSA and ECDSA encryption algorithms and SHA1 and SHA256 for DSA.   <b>Note:</b> 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.
<b>Type</b>	enum
<b>Module</b>	N/crypto/certificate Module
<b>Sibling Module Members</b>	N/crypto/certificate Module Members
<b>Since</b>	2019.1

## Values

- SHA1
- SHA256
- SHA384
- SHA512

## N/currency Module

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

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 Samples

## N/currency Module Member

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">currency.exchangeRate(options)</a>	number	Client and server-side scripts	Returns an exchange rate between two currencies.

## N/currency Module Script Samples

The following script samples demonstrate how to use the features of the N/currency module.

### Sample 1: Obtain an exchange rate

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to obtain the exchange rate between the Canadian dollar and the U.S. dollar on a specific date.

```
/*
 * @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)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to return the exchange rate between two currencies based on a certain date.
---------------------------	---

	<p>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.</p> <p>The exchange rate values are sourced from the Currency Exchange Rate record.</p>
	<p><b>Note:</b> The Currency Exchange Rate record itself is not a scriptable record.</p>
<b>Returns</b>	The exchange rate as a decimal number
<b>Supported Script Types</b>	<p>Client and server-side scripts</p> <p>For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/currency Module</a>
<b>Since</b>	2015.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
options.date	Date	optional	<ul style="list-style-type: none"> <li>■ Pass in a new Date object. For example, <code>date: new Date('7/28/2015')</code></li> <li>■ If <code>date</code> is not specified, then it defaults to today (the current date).</li> <li>■ The date determines the exchange rate in effect. If there are multiple rates, it is the latest entry on that date.</li> <li>■ Use the same date format as your NetSuite account.</li> </ul>	2015.2
options.source	number   string	required	<ul style="list-style-type: none"> <li>■ The internal ID or three-letter ISO code for the currency you are converting from.</li> <li>■ For example, you can use either <b>1</b> (internal ID) or <b>USD</b> (currency code).</li> <li>■ 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 <a href="#">Lists &gt; Accounting &gt; Currencies</a>.</li> </ul>	2015.2
options.target	number   string	required	<ul style="list-style-type: none"> <li>■ The internal ID or three-letter ISO code for the currency you are converting to.</li> </ul>	2015.2

## Errors

Error Code	Message	Thrown If
MISSING_REQD_ARGUMENT	exchangeRate: Missing a required argument: <source/target>	The <code>source</code> or <code>target</code> argument is missing.

Error Code	Message	Thrown If
SSS_INVALID_CURRENCY_ID	You have entered an invalid currency symbol or internal ID: <target/source>	The <code>source</code> or <code>target</code> 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 > Accounting > Currencies.

## Syntax

 **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.

```
//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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

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 Samples.

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

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

## Column Object Members

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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	<code>Column.id</code>	string (read-only)	Client scripts	Returns the internal ID of the column.
	<code>Column.label</code>	string (read-only)	Client scripts	Returns the UI label for the column.
	<code>Column.sublistId</code>	string (read-only)	Client scripts	Returns the internal ID of the standard or custom sublist that contains the column.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	Column.type	string (read-only)	Client scripts	Returns the column type.

## CurrentRecord Object Members

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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	CurrentRecord.cancelLine(options)	currentRecord.CurrentRecord	Client scripts	Cancels the changes made to the currently selected line.
	CurrentRecord.commitLine(options)	currentRecord.CurrentRecord	Client scripts	Commits the currently selected line.
	CurrentRecord.findMatrixSublistLineWithValue(options)	number	Client scripts	Returns the line number of the first line that contains the specified value in the matrix column.
	CurrentRecord.findSublistLineWithValue(options)	number	Client scripts	Gets the line number for the first occurrence of a field value in a sublist.
	CurrentRecord.getCurrentMatrixSublistValue(options)	number   Date   string   array   boolean <code>true</code>   <code>false</code>	Client scripts	Gets the value for the currently selected line in the matrix.
	CurrentRecord.getCurrentSublistIndex(options)	number	Client scripts	Gets the line number of the currently selected line.
	CurrentRecord.getCurrentSublistSubrecord(options)	currentRecord.CurrentRecord	Client scripts	Gets the subrecord for the associated sublist field on the current line. The subrecord object is retrieved in view mode.
	CurrentRecord.getCurrentSublistText(options)	number   Date   string   array   boolean <code>true</code>   <code>false</code>	Client scripts	Gets the value of the field in the currently selected line by text representation.
	CurrentRecord.getCurrentSublistValue(options)	number   Date   string   array   boolean <code>true</code>   <code>false</code>	Client scripts	Gets the value of the field in the currently selected line.
	CurrentRecord.getField(options)	currentRecord.Field	Client scripts	Gets a field object from the record.
	CurrentRecord.getLineCount(options)	number	Client scripts	Returns the number of lines in the sublist.
	CurrentRecord.getMatrixHeaderCount(options)	number	Client scripts	Returns the number of columns for the specified matrix.
	CurrentRecord.getMatrixHeaderField(options)	currentRecord.Field	Client scripts	Gets the field for the specified header in the matrix.
	CurrentRecord.getMatrixHeaderValue(options)	number   Date   string   array   boolean <code>true</code>   <code>false</code>	Client scripts	Gets the value for the associated header in the matrix.
	CurrentRecord.getMatrixSublistField(options)	currentRecord.Field	Client scripts	Gets the field for the specified sublist in the matrix.
	CurrentRecord.getMatrixSublistValue(options)	number   Date   string   array   boolean <code>true</code>   <code>false</code>	Client scripts	Gets the value for the associated field in the matrix.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	CurrentRecord.getSublist(options)	currentRecord.Sublist	Client scripts	Gets the specified sublist object.
	CurrentRecord.getSublistField(options)	currentRecord.Field	Client scripts	Gets the specified field object from the sublist.
	CurrentRecord.getSublistText(options)	string	Client scripts	Gets the value of the field in a sublist by a string representation.
	CurrentRecord.getSublistValue(options)	number   Date   string   array   boolean true   false	Client scripts	Gets the value of the field in a sublist.
	CurrentRecord.getSubrecord(options)	currentRecord.CurrentRecord	Client scripts	Gets the subrecord associated with the field. The subrecord object is retrieved in view mode.
	CurrentRecord.getText(options)	string	Client scripts	Gets the value of the field by a string representation.
	CurrentRecord.getValue(options)	number   Date   string   array   boolean true   false	Client scripts	Gets the value of the field.
	CurrentRecord.hasCurrentSublistSubrecord(options)	boolean true   false	Client scripts	Returns a value indicating whether the associated sublist field has a subrecord on the current line.
	CurrentRecord.hasSublistSubrecord(options)	boolean true   false	Client scripts	Returns a value indicating whether the associated sublist field contains a subrecord.
	CurrentRecord.hasSubrecord(options)	boolean true   false	Client scripts	Indicates whether the field has a subrecord.
	CurrentRecord.insertLine(options)	currentRecord.CurrentRecord	Client scripts	Inserts a new line in a sublist.
	CurrentRecord.removeCurrentSublistSubrecord(options)	currentRecord.CurrentRecord	Client scripts	Removes the subrecord for the associated sublist field on the current line.
	CurrentRecord.removeLine(options)	currentRecord.CurrentRecord	Client scripts	Removes a line from a sublist.
	CurrentRecord.removeSubrecord(options)	currentRecord.CurrentRecord	Client scripts	Removes the subrecord associated with the field.
	CurrentRecord.selectLine(options)	void	Client scripts	Selects a line item in a sublist.
	CurrentRecord.selectNewLine(options)	currentRecord.CurrentRecord	Client scripts	Selects a new line at the end of the sublist.
	CurrentRecord.setCurrentMatrixSublistValue(options)	currentRecord.CurrentRecord	Client scripts	Sets the value for the currently selected line in the matrix.
	CurrentRecord.setCurrentSublistText(options)	currentRecord.CurrentRecord	Client scripts	Sets the value of the field in the currently selected line using a string representation.
	CurrentRecord.setCurrentSublistValue(options)	currentRecord.CurrentRecord	Client scripts	Sets the value of the field in the currently selected line.
	CurrentRecord.setMatrixHeaderValue(options)	currentRecord.CurrentRecord	Client scripts	Sets the value for the associated header in the matrix.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	CurrentRecord.setMatrixSublistValue(options)	currentRecord. CurrentRecord	Client scripts	Sets the value for the associated field in the matrix.
	CurrentRecord.setText(options)	currentRecord. CurrentRecord	Client scripts	Sets the value of the field using a string representation.
	CurrentRecord.setValue(options)	currentRecord. CurrentRecord	Client scripts	Sets the value of the field.
Property	CurrentRecord.id	number (read-only)	Client scripts	Returns the internal record ID.
	CurrentRecord.isDynamic	boolean <b>true</b>   <b>false</b> (read-only)	Client scripts	Indicates whether the record is dynamic.
	CurrentRecord.type	string (read-only)	Client scripts	Returns the record type.

## Field Object Members

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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Field.getSelectOptions(options)	array	Client scripts	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.
	Field.insertSelectOption(options)	void	Client scripts	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.
	Field.removeSelectOption(options)	void	Client scripts	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..
Object	Field.id	string (read-only)	Client scripts	Returns the internal ID of a standard or custom body or sublist field.
	Field.isDisabled	boolean <b>true</b>   <b>false</b>	Client scripts	Returns <b>true</b> if the standard or custom field is disabled on the record form, or <b>false</b> otherwise.
	Field.isDisplay	boolean <b>true</b>   <b>false</b>	Client scripts	Returns <b>true</b> if the field is set to display on the record form, or <b>false</b> otherwise.  This property is read-only for sublist fields.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	Field.isMandatory	boolean <code>true</code>   <code>false</code>	Client scripts	Returns <code>true</code> if the standard or custom field is mandatory on the record form, or <code>false</code> otherwise.
	Field.isPopup	boolean <code>true</code>   <code>false</code> (read-only)	Client scripts	Returns <code>true</code> if the field is a popup list field, or <code>false</code> otherwise.
	Field.isReadOnly	boolean <code>true</code>   <code>false</code>	Client scripts	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.
	Field.isVisible	boolean <code>true</code>   <code>false</code> (read-only)	Client scripts	Returns <code>true</code> if the field is visible on the record form, or <code>false</code> otherwise.
	Field.label	string (read-only)	Client scripts	Returns the UI label for a standard or custom field body or sublist field.
	Field.sublistId	string (read-only)	Client scripts	Returns the ID of the sublist associated with the specified sublist field.
	Field.type	string (read-only)	Client scripts	Returns the type of a body or sublist field.

## Sublist Object Members

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

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

## N/currentRecord Module Script Sample

**Note:** This script sample uses the `define` function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the `require` function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample 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.

```
/*
 * @NApiVersion 2.0
 */

define(['N/currentRecord'], function(currentRecord) {
    return({
        test_set_getValue: function() {
            var record = currentRecord.get();
            record.setValue({
                fieldId: 'custpage_textfield',
                value: 'Body value',
                ignoreFieldChange: true,
                forceSyncSourcing: true
            });
            var actValue = record.getValue({
                fieldId: 'custpage_textfield'
            });
            record.setValue({
                fieldId: 'custpage_resultfield',
                value: actValue,
                ignoreFieldChange: true,
                forceSyncSourcing: true
            });
        },
        test_set_getCurrentSublistValue: function() {
            var record = currentRecord.get();
            record.setCurrentSublistValue({
                sublistId: 'itemvendor',
                fieldId: 'custpage_subtextfield',
                value: 'Sublist Value',
                ignoreFieldChange: true,
                forceSyncSourcing: true
            });
            var actValue = record.getCurrentSublistValue({
                sublistId: 'itemvendor',
                fieldId: 'custpage_subtextfield'
            });
            record.setValue({
                fieldId: 'custpage_sublist_resultfield',
                value: actValue
            });
        }
    });
})
```

```

        value: actValue,
        ignoreFieldChange: true,
        forceSyncSourcing: true
    );
},
);
});

```

**i Note:** This script sample uses the **define** function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the **require** function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample 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 sample is to set up a page, adding fields and buttons, so that you can use the code you made in the first sample, and see the fields and buttons in action.

```

/**
 * @NApiVersion 2.0
 * @NScriptType UserEventScript
 * @NModuleScope SameAccount
 */

define([], function() {
    return {
        beforeLoad: function (params) {
            {
                var form = params.form;

                var textfield = form.addField({
                    id: 'custpage_textfield',
                    type: 'text',
                    label: 'Text'
                });
                var resultfield = form.addField({
                    id: 'custpage_resultfield',
                    type: 'text',
                    label: 'Result'
                });
                var sublistResultfield = form.addField({
                    id: 'custpage_sublist_resultfield',
                    type: 'text',
                    label: 'Sublist Result Field'
                });

                var sublistObj = form.getSublist({
                    id: 'itemvendor'
                });
                var subtextfield = sublistObj.addField({
                    id: 'custpage_subtextfield',
                    type: 'text',
                    label: 'Sublist Text Field'
                });
            }
        }
    };
});

```

```

        form.clientScriptModulePath = './clientDemo.js';
        form.addButton({
            id: 'custpage_custombutton',
            label: 'SET_GET_VALUE',
            functionName: 'test_set_getValue'
        });
        form.addButton({
            id: 'custpage_custombutton2',
            label: 'SET_GETCURRENTSUBLISTVALUE',
            functionName: 'test_set_getCurrentSublistValue'
        });
    }
};

});

```

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following samples shows how to use the `forceSyncSourcing` parameter.

```

/**
 * @NApiVersion 2.x
 */

require(['N/currentRecord'], function(currentRecord){
    var rec = currentRecord.get();
    rec.selectNewLine({sublistId: 'item'});
    rec.setCurrentSublistValue({
        sublistId: 'item',
        fieldId: 'item',
        value: 39,
        forceSyncSourcing:true
    });
    rec.setCurrentSublistValue({
        sublistId: 'item',
        fieldId: 'quantity',
        value: 1,
        forceSyncSourcing:true
    });
    rec.commitLine({sublistId: 'item'});
})

```

## currentRecord.Column

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a column of a sublist on the current record. For a complete list of this object's properties, see <a href="#">Column Object Members</a> .
<b>Supported Script Types</b>	Client scripts

	For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	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](#).

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

## Column.id

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the internal ID of the column.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	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](#).

```
//Add additional code ...
var columnid = objColumn.id;
...
//Add additional code
```

## Column.label

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the internal ID of the column.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client scripts

	For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	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](#).

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

## Column.sublistId

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the internal ID of the standard or custom sublist that contains the column.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	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](#).

```
//Add additional code
...
var sublistid = objColumn.sublistId;
...
//Add additional code
```

## Column.type

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the column type.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client scripts

	For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	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](#).

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

## currentRecord.CurrentRecord

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the record active on the current page.
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	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.

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

In an entry point client script, you do not have 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.

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

```

function pageInit(context) {
    var currentRec = context.currentRecord;
    ...
//Add additional code

```

## CurrentRecord.cancelLine(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Cancels the currently selected line on a sublist.
<b>Returns</b>	The <a href="#">currentRecord.CurrentRecord</a> object that called the method.
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2

### Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

### Syntax

 **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](#).

```

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

```

```
//Add additional code
```

## CurrentRecord.commitLine(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Commits the currently selected line on a sublist.
<b>Returns</b>	The <a href="#">currentRecord.CurrentRecord</a> object that called the method.
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
options.ignoreRecalc	boolean true   false	optional	If set to true, scripting recalculation is ignored.	2016.2

### Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

### Syntax

**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](#).

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

```
//Add additional code
```

## CurrentRecord.findMatrixSublistLineWithValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the line number of the first instance where a specified value is found in a specified column of the matrix.
<b>Returns</b>	number
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
options.fieldId	string	required	The ID of the matrix field.  See the help topic <a href="#">How do I find a field's internal ID?</a>	2016.2
options.value	number	required	The value to search for.	2016.2
options.column	number	required	The column number of the field.	2016.2

### Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

### Syntax

 **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](#).

```
//Add additional code
```

```

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

```

## CurrentRecord.findSublistLineWithValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the line number for the first occurrence of a field value in a sublist.
<b>Returns</b>	A line number as a number, or -1 if not found.
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	N/currentRecord Module
<b>Since</b>	2016.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See the help topic <a href="#">How do I find a field's internal ID?</a>	2016.2
options.value	number   Date   string   array   boolean true   false	optional	The value to search for.	2016.2

### Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or not defined.

## Syntax



**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](#).

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

## CurrentRecord.getCurrentMatrixSublistValue(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the value for the currently selected line in the matrix. Gets a numeric value for rate and ratehighprecision fields.
<b>Returns</b>	number   Date   string   array   boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	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 <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
options.fieldId	string	required	The internal ID of the matrix field.  See the help topic <a href="#">How do I find a field's internal ID?</a>	2016.2
options.column	number	required	The column number for the matrix field.	2016.2

## Errors

Error Code	Thrown If
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 script example, see [N/currentRecord Module Script Sample](#).

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

## CurrentRecord.getCurrentSublistIndex(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the line number of the currently selected line.
<b>Returns</b>	number
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2

## Errors

Error Code	Thrown If
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 script example, see [N/currentRecord Module Script Sample](#).

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

## CurrentRecord.getCurrentSublistSubrecord(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the subrecord for the associated sublist field on the current line. The subrecord object is retrieved in view mode.
<b>Returns</b>	<code>currentRecord.CurrentRecord</code>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	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](#).

Parameter	Type	Required / Optional	Description	Since
<code>options.sublistId</code>	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
<code>options.fieldId</code>	string	required	The internal ID of a standard or custom sublist field.	2016.2

Parameter	Type	Required / Optional	Description	Since
			See the help topic <a href="#">How do I find a field's internal ID?</a>	

## Syntax

 **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](#).

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

## CurrentRecord.getCurrentSublistText(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a text representation of the field value in the currently selected line.
<b>Returns</b>	string   <b>Note:</b> For multiselect fields, returns an array.
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	2016.2

Parameter	Type	Required / Optional	Description	Since
			See the help topic <a href="#">How do I find a field's internal ID?</a>	

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax

 **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](#).

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

## CurrentRecord.getCurrentSublistValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the value of a sublist field on the currently selected sublist line.
<b>Returns</b>	number   Date   string   array   boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.	2016.2

Parameter	Type	Required / Optional	Description	Since
			This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See the help topic <a href="#">How do I find a field's internal ID?</a>	2016.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax



**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](#).

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

## CurrentRecord.getField(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a field object from a record.
<b>Returns</b>	<code>currentRecord.Field</code>
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	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>	2016.2

## Errors

Error Code	Thrown If
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 script example, see [N/currentRecord Module Script Sample](#).

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

## CurrentRecord.getLineCount(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the number of lines in a sublist.
<b>Returns</b>	number
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	<p>The internal ID of the sublist.</p> <p>This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a>.</p>	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](#).

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

## CurrentRecord.getMatrixHeaderCount(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the number of columns for the specified matrix.
<b>Returns</b>	number
<b>Supported Script Types</b>	<p>Client scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a>.</p>
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	<p>The internal ID of the sublist that contains the matrix.</p> <p>This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a>.</p>	2016.2

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	The internal ID of the matrix field.  See the help topic <a href="#">How do I find a field's internal ID?</a>	2016.2

## Errors

Error Code	Thrown If
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 script example, see [N/currentRecord Module Script Sample](#).

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

## CurrentRecord.getMatrixHeaderField(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the field for the specified header in the matrix.
<b>Returns</b>	<a href="#">currentRecord.Field</a>
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist that contains the matrix.	2016.2

Parameter	Type	Required / Optional	Description	Since
			This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	
options.fieldId	string	required	The internal ID of the matrix field.  See the help topic <a href="#">How do I find a field's internal ID?</a>	2016.2
options.column	number	required	The column number for the field.	2016.2

## Errors

Error Code	Thrown If
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 script example, see [N/currentRecord Module Script Sample](#).

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

## CurrentRecord.getMatrixHeaderValue(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the value for the associated header in the matrix.
<b>Returns</b>	number   Date   string   array   boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	N/currentRecord Module
<b>Since</b>	2016.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	<p>The internal ID of the sublist that contains the matrix.</p> <p>This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a>.</p>	2016.2
options.fieldId	string	required	<p>The internal ID of the matrix field.</p> <p>See the help topic <a href="#">How do I find a field's internal ID?</a></p>	2016.2
options.column	number	required	The column number for the field.	2016.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

## Syntax

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

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

## CurrentRecord.getMatrixSublistField(options)

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Gets the field for the specified sublist in the matrix.
<b>Returns</b>	<code>currentRecord.Field</code>
<b>Supported Script Types</b>	<p>Client scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a>.</p>
<b>Governance</b>	None

<b>Module</b>	N/currentRecord Module
<b>Since</b>	2016.2

## Parameters

<b>i</b>	<b>Note:</b> The options parameter is a JavaScript object.
----------	--

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	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 <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
options.fieldId	string	required	The internal ID of the matrix field.  See the help topic <a href="#">How do I find a field's internal ID?</a>	2016.2
options.column	number	required	The column number for the field.	2016.2
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	2016.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

## Syntax

<b>⚠ Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/currentRecord Module Script Sample</a> .
--

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

## CurrentRecord.getMatrixSublistValue(options)

<b>i</b>	<b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.
----------	--

<b>Method Description</b>	Gets the value for the associated field in the matrix.
---------------------------	--

<b>Returns</b>	number   Date   string   array   boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

<b>Note:</b>	The options parameter is a JavaScript object.
--------------	---

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	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 <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
options.fieldId	string	required	The internal ID of the matrix field.  See the help topic <a href="#">How do I find a field's internal ID?</a>	2016.2
options.column	number	required	The column number for the field.	2016.2
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	2016.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

## Syntax

<b>Important:</b>	The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/currentRecord Module Script Sample</a> .
-------------------	--

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

```
//Add additional code
```

## CurrentRecord.getSublist(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the specified sublist.
<b>Returns</b>	<code>currentRecord.Sublist</code>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.sublistId</code>	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	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](#).

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

## CurrentRecord.getSublistField(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a field object from a sublist.
---------------------------	--

<b>Returns</b>	currentRecord.Field
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

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

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax

 <b>Important:</b>	The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/currentRecord Module Script Sample</a> .
---	--

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

## CurrentRecord.getSublistText(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the value of a sublist field in a text representation. Gets a string value with a "%" for rate and ratehighprecision fields.
<b>Returns</b>	string   <b>Note:</b> For multiselect fields, returns an array.
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

### Parameters

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

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

### Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

### Syntax

 **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](#).

```
//Add additional code
```

```

...
var sublistFieldName = objRecord.getSublistText({
    sublistId: 'item',
    fieldId: 'item',
    line: 3
});
...
//Add additional code

```

## CurrentRecord.getSublistValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the value of a sublist field.  Gets a numeric value for rate and ratehighprecision fields.
<b>Returns</b>	number   Date   string   array   boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

### Parameters

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

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

### Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax

**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](#).

```
//Add additional code
...
var quantity = record.getSublistValue({
    sublistId: 'item',
    fieldId: 'quantity',
    line: 0
});
...
//Add additional code
```

## CurrentRecord.getSubrecord(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the subrecord associated with the field. The subrecord object is available in view mode.
<b>Returns</b>	<code>currentRecord.CurrentRecord</code>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.fieldId</code>	string	required	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>	2016.2

## Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	A required argument is missing or undefined.

Error Code	Thrown If
FIELD_1_IS_NOT_A_SUBRECORD_FIELD	The specified field is not a subrecord field.
FIELD_1_IS_DISABLED_YOU_CANNOT_APPLY_SUBRECORD_OPERATION_ON_THIS_FIELD	The specified field is disabled.
SSS_INVALID_FIELD_ON_SUBRECORD_OPERATION	The specified fieldId does not refer to a subrecord.

## Syntax

 **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](#).

```
//Add additional code
...
var sublistFieldValue = objRecord.getSubrecord({
    fieldId: 'subrecord'
});
...
//Add additional code
```

## CurrentRecord.getText(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the text representation of a field value. Gets a string value with a "%" for rate and ratehighprecision fields.
<b>Returns</b>	string   <b>Note:</b> For multiselect fields, returns an array.
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	The internal ID of a standard or custom body field.	2016.2

Parameter	Type	Required / Optional	Description	Since
			See the help topic <a href="#">How do I find a field's internal ID?</a>	

## Errors

Error Code	Thrown If
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 script example, see [N/currentRecord Module Script Sample](#).

```
//Add additional code
...
var fieldidname = objRecord.getText({
    fieldId: 'item'
});
...
//Add additional code
```

## CurrentRecord.getValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the value of a field.
<b>Returns</b>	number   Date   string   array   boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	N/currentRecord Module
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	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>	2016.2

## Errors

Error Code	Thrown If
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 script example, see [N/currentRecord Module Script Sample](#).

```
//Add additional code
...
var value = objRecord.getValue({
    fieldId: 'item'
});
...
//Add additional code
```

## CurrentRecord.hasCurrentSublistSubrecord(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	boolean <code>true</code>   <code>false</code>
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.sublistId</code>	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	The internal ID of a subrecord.  See the help topic <a href="#">How do I find a field's internal ID?</a>	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](#).

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

## CurrentRecord.hasSublistSubrecord(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a value indicating whether the associated sublist field contains a subrecord.
<b>Returns</b>	boolean <code>true</code>   <code>false</code>
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
options.fieldId	string	required	The internal ID of a subrecord.	2016.2

Parameter	Type	Required / Optional	Description	Since
			See the help topic <a href="#">How do I find a field's internal ID?</a>	
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	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](#).

```
//Add additional code
...
var hasSubrecord = objRecord.hasSublistSubrecord({
    sublistId: 'item',
    fieldId: 'item',
    line: 3
});
...
//Add additional code
```

## CurrentRecord.hasSubrecord(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a value indicating whether the field contains a subrecord.
<b>Returns</b>	boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	The internal ID of the field that may contain a subrecord.  See the help topic <a href="#">How do I find a field's internal ID?</a>	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](#).

```
//Add additional code
...
var hasSubrecord = objRecord.hasSubrecord({
    fieldId: 'item'
});
...
//Add additional code
```

## CurrentRecord.insertLine(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Inserts a sublist line.
<b>Returns</b>	<a href="#">currentRecord.CurrentRecord</a>
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
options.line	number	required	The line number to insert. Note that line indexing begins at 0 with SuiteScript 2.0.	2016.2
options.ignoreRecalc	boolean <code>true</code>   <code>false</code>	optional	If set to true, scripting recalculation is ignored.  The default value is false.	2016.2

## Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	A required argument is missing or undefined.

Error Code	Thrown If
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax

 **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](#).

```
//Add additional code
...
objRecord.insertLine({
    sublistId: 'item',
    line: 3,
    ignoreRecalc: true
});
...
//Add additional code
```

## CurrentRecord.removeCurrentSublistSubrecord(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Removes the subrecord for the associated sublist field.
<b>Returns</b>	<code>currentRecord.CurrentRecord</code>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.sublistId</code>	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
<code>options.fieldId</code>	string	required	The internal ID of a standard or custom sublist field.  See the help topic <a href="#">How do I find a field's internal ID?</a>	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](#).

```
//Add additional code
...
objRecord.removeCurrentSublistSubrecord({
    sublistId: 'item',
    fieldId: 'item'
});
...
//Add additional code
```

## CurrentRecord.removeLine(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Removes a sublist line.
<b>Returns</b>	<code>currentRecord.CurrentRecord</code>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.sublistId</code>	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
<code>options.line</code>	number	required	The line number of the sublist to remove. Note that line indexing begins at 0 with SuiteScript 2.0.	2016.2
<code>options.ignoreRecalc</code>	boolean <code>true</code>   <code>false</code>	optional	If set to true, scripting recalculation is ignored.  The default value is false.	2016.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax

**⚠ 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](#).

```
//Add additional code
...
objRecord.removeLine({
    sublistId: 'item',
    line: 3,
    ignoreRecalc: true
});
...
//Add additional code
```

## CurrentRecord.removeSubrecord(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Removes the subrecord for the associated field.
<b>Returns</b>	<a href="#">currentRecord.CurrentRecord</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	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>	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](#).

```
//Add additional code
...
objRecord.removeSubrecord({
    fieldid: 'item'
});
...
//Add additional code
```

## CurrentRecord.selectLine(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Selects an existing line in a sublist.
<b>Returns</b>	<a href="#">currentRecord.CurrentRecord</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
options.line	number	required	The line number to select in the sublist. Note that line indexing begins at 0 with SuiteScript 2.0.	2016.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax

**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](#).

```
//Add additional code
...
var record = CurrentRecord.selectLine({
    sublistId: 'item',
    line: 3
});
...
//Add additional code
```

## CurrentRecord.selectNewLine(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Selects a new line at the end of a sublist.
<b>Returns</b>	<a href="#">currentRecord.CurrentRecord</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax



**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](#).

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

## CurrentRecord.setCurrentMatrixSublistValue(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	<a href="#">currentRecord.CurrentRecord</a>
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	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 <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
options.fieldId	string	required	The internal ID of the matrix field.  See the help topic <a href="#">How do I find a field's internal ID?</a>	2016.2
options.column	number	required	The column number for the field.	2016.2

Parameter	Type	Required / Optional	Description	Since
options.value	number   Date   string   array   boolean <b>true</b>   <b>false</b>	required	<p>The value to set the field to.</p> <p>The value type must correspond to the field type being set. For example:</p> <ul style="list-style-type: none"> <li>■ Text, Radio and Select fields accept string values.</li> <li>■ Checkbox fields accept Boolean values.</li> <li>■ Date and DateTime fields accept Date values.</li> <li>■ Integer, Float, Currency and Percent fields accept number values.</li> </ul>	2016.2
options.ignoreFieldChange	boolean <b>true</b>   <b>false</b>	optional	<p>If set to <b>true</b>, the field change and slaving event is ignored.</p> <p>By default, this value is <b>false</b>.</p>	2016.2
options.forceSyncSourcing	boolean <b>true</b>   <b>false</b>	optional	<p>Indicates whether to perform field sourcing synchronously.</p> <p>If set to <b>true</b>, sources dependent field information for empty fields synchronously.</p> <p>Defaults to <b>false</b> – dependent field values are not sourced synchronously.</p>	2019.1

## Errors

Error Code	Thrown If
INVALID_FLD_VALUE	The options.value type does not match the field type.
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 script example, see [N/currentRecord Module Script Sample](#).

```
//Add additional code
...
objRecord.setCurrentMatrixSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 3,
    value: false,
    ignoreFieldChange: true,
    forceSyncSourcing: true
});
...
//Add additional code
```

## CurrentRecord.setCurrentSublistText(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	<code>currentRecord.CurrentRecord</code>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.sublistId</code>	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
<code>options.fieldId</code>	string	required	The internal ID of a standard or custom sublist field.  See the help topic <a href="#">How do I find a field's internal ID?</a>	2016.2
<code>options.text</code>	string	required	The text to set the value to.	2016.2
<code>options.ignoreFieldChange</code>	boolean <code>true</code>   <code>false</code>	optional	If set to <code>true</code> , the field change and slaving event is ignored.  By default, this value is <code>false</code> .	2016.2
<code>options.forceSyncSourcing</code>	boolean <code>true</code>   <code>false</code>	optional	Indicates whether to perform field sourcing synchronously.  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.	2019.1

### Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	A required argument is missing or undefined.

Error Code	Thrown If
A_SCRIPT_IS_ATTEMPTING_TO_EDIT_THE_1_SUBLIST_THIS_SUBLIST_IS_CURRENTLY_IN_READONLY_MODE_AND_CANNOT_BE_EDITED_CALL_YOUR_NETSUITE_ADMINISTRATOR_TO_DISABLE_THIS_SCRIPT_IF_YOU_NEED_TO_SUBMIT_THIS_RECORD	A user tries to edit a read-only sublist 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/currentRecord Module Script Sample](#).

```
//Add additional code
...
objRecord.setCurrentSublistText({
    sublistId: 'item',
    fieldId: 'item',
    text: 'value',
    ignoreFieldChange: true,
    forceSyncSourcing: true
});
...
//Add additional code
```

## CurrentRecord.setCurrentSublistValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value for the field in the currently selected line.   <b>Important:</b> 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.
<b>Returns</b>	<code>currentRecord.CurrentRecord</code>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.sublistId</code>	string	required	The internal ID of the sublist.	2016.2

Parameter	Type	Required / Optional	Description	Since
			This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See the help topic <a href="#">How do I find a field's internal ID?</a>	2016.2
options.value	number   Date   string   array   boolean <b>true</b>   <b>false</b>	required	The value to set the field to.  The value type must correspond to the field type being set. For example: <ul style="list-style-type: none"><li>■ Text, Radio and Select fields accept string values.</li><li>■ Checkbox fields accept Boolean values.</li><li>■ Date and DateTime fields accept Date values.</li><li>■ Integer, Float, Currency and Percent fields accept number values.</li></ul>	2016.2
options.ignoreFieldChange	boolean <b>true</b>   <b>false</b>	optional	If set to <b>true</b> , the field change and slaving event is ignored.  By default, this value is <b>false</b> .	2016.2
options.forceSyncSourcing	boolean <b>true</b>   <b>false</b>	optional	Indicates whether to perform field sourcing synchronously.  By default, this value is <b>false</b> .	2019.1

## Errors

Error Code	Thrown If
INVALID_FLD_VALUE	The <b>options.value</b> type does not match the field type.
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
A_SCRIPT_IS_ATTEMPTING_TO_EDIT_THE_1_SUBLIST_THIS_SUBLIST_IS_CURRENTLY_IN_READONLY_MODE_AND_CANNOT_BE_EDITED_CALL_YOUR_NETSUITE_ADMINISTRATOR_TO_DISABLE_THIS_SCRIPT_IF_YOU_NEED_TO_SUBMIT_THIS_RECORD	A user tries to edit a read-only sublist 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/currentRecord Module Script Sample](#).

```
//Add additional code
...
objRecord.setCurrentSublistValue({
    sublistId: 'item',
```

```

    fieldId: 'item',
    value: true,
    ignoreFieldChange: true
});
...
//Add additional code

```

## CurrentRecord.setMatrixHeaderValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value for the associated header in the matrix.  Sets a numeric value for rate and ratehighprecision fields.
<b>Returns</b>	<code>currentRecord.CurrentRecord</code>
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.sublistId</code>	string	required	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 <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
<code>options.fieldId</code>	string	required	The internal ID of the matrix field.  See the help topic <a href="#">How do I find a field's internal ID?</a>	2016.2
<code>options.column</code>	number	required	The column number for the field.	2016.2
<code>options.value</code>	number   Date   string   array   boolean <code>true</code>   <code>false</code>	required	The value to set the field to.  The value type must correspond to the field type being set. For example: <ul style="list-style-type: none"><li>■ Text, Radio and Select fields accept string values.</li><li>■ Checkbox fields accept Boolean values.</li></ul>	2016.2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> <li>■ Date and DateTime fields accept Date values.</li> <li>■ Integer, Float, Currency and Percent fields accept number values.</li> </ul>	
options.ignoreFieldChange	boolean true   false	optional	If set to <b>true</b> , the field change and slaving event is ignored.  By default, this value is <b>false</b> .	2016.2
options.forceSyncSourcing	boolean true   false	optional	Indicates whether to perform field sourcing synchronously.  If set to <b>true</b> , sources dependent field information for empty fields synchronously.  Defaults to <b>false</b> – dependent field values are not sourced synchronously.	2019.1

## Errors

Error Code	Thrown If
INVALID_FLD_VALUE	The <b>options.value</b> type does not match the field type.
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 script example, see [N/currentRecord Module Script Sample](#).

```
//Add additional code
...
objRecord.setMatrixHeaderValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 3,
    value: false,
    ignoreFieldChange: true,
    forceSyncSourcing: true
});
...
//Add additional code
```

## CurrentRecord.setMatrixSublistValue(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value for the associated field in the matrix.  Sets a numeric value for rate and ratehighprecision fields.
---------------------------	---

<b>Returns</b>	currentRecord.CurrentRecord
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	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 <a href="#">Working with the SuiteScript Records Browser</a> .	2016.2
options.fieldId	string	required	The internal ID of the matrix field.  See the help topic <a href="#">How do I find a field's internal ID?</a>	2016.2
options.column	number	required	The column number for the field.	2016.2
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	2016.2
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: <ul style="list-style-type: none"><li>■ Text, Radio and Select fields accept string values.</li><li>■ Checkbox fields accept Boolean values.</li><li>■ Date and DateTime fields accept Date values.</li><li>■ Integer, Float, Currency and Percent fields accept number values.</li></ul>	2016.2

## Errors

Error Code	Thrown If
INVALID_FLD_VALUE	The <code>options.value</code> type does not match the field type.
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 script example, see [N/currentRecord Module Script Sample](#).

```
//Add additional code
...
objRecord.setMatrixSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 12,
    line: 3,
    value: true
});
...
//Add additional code
```

## CurrentRecord.setText(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value of the field by a text representation.  Sets a string value with a "%" for rate and ratehighprecision fields.
<b>Returns</b>	<a href="#">currentRecord.CurrentRecord</a>
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	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>	2016.2
options.text	string	required	The text to change the field value to.	2016.2
options.ignoreFieldChange	boolean <code>true</code>   <code>false</code>	optional	If set to <code>true</code> , the field change and slaving event is ignored.	2016.2

Parameter	Type	Required / Optional	Description	Since
			By default, this value is <b>false</b> .	
options. forceSyncSourcing	boolean <b>true</b>   <b>false</b>	optional	Indicates whether to perform field sourcing synchronously.  If set to <b>true</b> , sources dependent field information for empty fields synchronously.  Defaults to <b>false</b> – dependent field values are not sourced synchronously.	2019.1

## Errors

Error Code	Thrown If
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 script example, see [N/currentRecord Module Script Sample](#).

```
//Add additional code
...
objRecord.setText({
    fieldId: 'item',
    text: 'value',
    ignoreFieldChange: true,
    forceSyncSourcing: true
});
...
//Add additional code
```

## CurrentRecord.setValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value of a field.  Sets a numeric value for rate and ratehighprecision fields.
<b>Returns</b>	<a href="#">currentRecord.CurrentRecord</a>
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	<p>The internal ID of a standard or custom body field.</p> <p>See the help topic <a href="#">How do I find a field's internal ID?</a></p>	2016.2
options.value	number   Date   string   array   boolean <b>true</b>   <b>false</b>	required	<p>The value to set the field to.</p> <p>The value type must correspond to the field type being set. For example:</p> <ul style="list-style-type: none"> <li>■ Text, Radio, Select and Multi-Select fields accept string values.</li> <li>■ Checkbox fields accept Boolean values.</li> <li>■ Date and DateTime fields accept Date values.</li> <li>■ Integer, Float, Currency and Percent fields accept number values.</li> </ul>	2016.2
options.ignoreFieldChange	boolean <b>true</b>   <b>false</b>	optional	<p>If set to <b>true</b>, the field change and slaving event is ignored.</p> <p>By default, this value is <b>false</b>.</p>	2016.2
options.forceSyncSourcing	boolean <b>true</b>   <b>false</b>	optional	<p>Indicates whether to perform field sourcing synchronously.</p> <p>If set to <b>true</b>, sources dependent field information for empty fields synchronously.</p> <p>Defaults to <b>false</b> – dependent field values are not sourced synchronously.</p>	2019.1

## Errors

Error Code	Thrown If
INVALID_FLD_VALUE	The <b>options.value</b> type does not match the field type.
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

## Syntax

 <b>Important:</b>	The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/currentRecord Module Script Sample</a> .
---	--

```
//Add additional code
```

```

...
objRecord.setValue({
    fieldId: 'item',
    value: true,
    ignoreFieldChange: true,
    forceSyncSourcing: true
});
...
//Add additional code

```

## CurrentRecord.id



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal ID of a specific record.
<b>Type</b>	number (read-only)
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	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](#).

```

//Add additional code
...
var recordid = record.id;
...
//Add additional code

```

## CurrentRecord.isDynamic



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether the record is in dynamic mode. For more information, see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> .  This value is set when the record is created or accessed.
<b>Type</b>	boolean <code>true</code>   <code>false</code> (read-only)
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	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](#).

```
//Add additional code
...
if (record.isDynamic) {
    ...
}
...
//Add additional code
```

## CurrentRecord.type



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The current record's type.  Note the following: <ul style="list-style-type: none"><li>■ On an instance of a standard record type, this property is represented by a value from the <code>record.Type</code> enum.</li><li>■ 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 <a href="#">Custom Record</a>.</li></ul>
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	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](#).

```
//Add additional code
...
var recordtype = currentRecord.type;
...
//Add additional code
```

## currentRecord.Field



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a body or sublist field on the current record.
---------------------------	---

	<p>Use the following methods to access the Field object:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">CurrentRecord.getField(options)</a></li> <li>▪ <a href="#">CurrentRecord.getSublistField(options)</a></li> </ul> <p>For a complete list of this object's methods and properties, see <a href="#">N/currentRecord Module</a>.</p>
<b>Supported Script Types</b>	<p>Client scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a>.</p>
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	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](#).

```
//Add additional code
...
var currentRecordField = currentRecord.getField({
    fieldId: 'entity'
});
...
//Add additional code
}
```

## Field.getSelectOptions(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Returns an array of available options on a standard or custom select, multiselect, or radio field as key-value pairs.</p> <p><b>Important:</b> You can use this method only in dynamic mode. For additional information on dynamic mode, see <a href="#">CurrentRecord.isDynamic</a>.</p>
<b>Returns</b>	<p>array</p> <p>Only the first 1,000 available options are returned in an array.</p> <p>If there are more than 1,000 available options, an empty array [] is returned.</p> <p>This function returns an array in the following format:</p> <pre>[{value: 5, text: 'abc'}, {value: 6, text: '123'}]</pre> <p>This function returns Type <b>Error</b> if the field is not a supported field for this method.</p>
<b>Governance</b>	None
<b>Supported Script Types</b>	<p>Client scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a>.</p>

<b>Module</b>	N/currentRecord Module
<b>Since</b>	2016.2

## Parameters

<b>i Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
options.filter	string	Required	The search string to filter the select options that are returned.  <b>i Note:</b> Filter values are case insensitive.	2016.2
options.operator	string	Required	The following operators are supported: <ul style="list-style-type: none"><li>■ <b>contains</b> (default)</li><li>■ <b>is</b></li><li>■ <b>startswith</b></li></ul>	2016.2

## Syntax

<b>⚠ Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/currentRecord Module Script Sample</a> .
--

```
//Add additional code
...
var options = objField.getSelectOptions({
    filter : 'C',
    operator : 'startswith'
});
...
//Add additional code
```

## Field.insertSelectOption(options)

<b>i Note:</b> The content in this help topic pertains to SuiteScript 2.0.
--

<b>Method Description</b>	Inserts an option into certain types of select and multiselect fields.  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 <b>custpage</b> .
<b>Returns</b>	Void
<b>Governance</b>	None
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .

<b>Module</b>	N/currentRecord Module
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.value	string	Required	A string, not visible in the UI, that identifies the option.	2016.2
options.text	string	Required	The label that represents the option in the UI.	2016.2
options.isSelected	boolean	Optional	Determines whether the option is selected by default. If not specified, this value defaults to false.	2016.2

## 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 <b>custpage</b> .

## Syntax

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

```
//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'
});
...
//Add additional code
```

## Field.removeSelectOption(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <b>custpage</b> .
<b>Returns</b>	Void
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.value	string	Required	A string, not shown in the UI, that identifies the option.  To remove all options from the list, set this field to null, as follows:  <pre>... field.removeSelectOption({     value: null, }); ...</pre>	2016.2

### 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 <b>custpage</b> .

### Syntax



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

```
//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',
});

...
//Add additional code
```

## Field.id



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the internal ID of a standard or custom body or sublist field.
<b>Type</b>	string (read-only)
<b>Module</b>	N/currentRecord Module
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Since</b>	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](#).

```
//Add additional code
...
var id = objField.id;
...
//Add additional code
```

## Field.label



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the UI label for a standard or custom field body or sublist field.
<b>Type</b>	string (read-only)
<b>Module</b>	N/currentRecord Module

<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Since</b>	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](#).

```
//Add additional code
...
var label = objField.label;
...
//Add additional code
```

## Field.isMandatory

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns <b>true</b> if the standard or custom field is mandatory on the record form, or <b>false</b> otherwise.
<b>Type</b>	boolean <b>true</b>   <b>false</b>
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Since</b>	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](#).

```
//Add additional code
...
if (objField.isMandatory) {
    ...
}
...
//Add additional code
```

## Field.isDisabled

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	This property reflects the display type of a field. A value of <b>true</b> means the field is disabled. A value of <b>false</b> means the field is enabled. Note also:
-----------------------------	--

	<ul style="list-style-type: none"> <li>■ If you are working with a body field, you can use this property to change the field's display type.</li> <li>■ If you are working with a sublist field, you can set this property to <code>true</code> or <code>false</code>, but be aware that this action affects the entire sublist column, even though a sublist field is associated with one line.</li> <li>■ For both body and sublist fields, you can use <code>Field.isDisabled</code> to determine whether the field is disabled or enabled.</li> </ul>
<b>Type</b>	boolean <code>true</code>   <code>false</code>
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Since</b>	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](#).

```
//Add additional code
...
if (objField.isDisabled) {
  ...
}
...
//Add additional code
```

## Field.isPopup



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns <code>true</code> if the field is a popup list field, or <code>false</code> otherwise.
<b>Type</b>	boolean <code>true</code>   <code>false</code> (read-only)
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Since</b>	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](#).

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

```
if (objField.isPopup) {
    ...
}
...
//Add additional code
```

## Field.isDisplay



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns <b>true</b> if the field is set to display on the record form, or <b>false</b> 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.
<b>Type</b>	boolean <b>true</b>   <b>false</b>
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Since</b>	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](#).

```
//Add additional code
...
if (objField.isDisplay) {
    ...
}
...
//Add additional code
```

## Field.isVisible



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns <b>true</b> if the field is visible on the record form, or <b>false</b> otherwise.
<b>Type</b>	boolean <b>true</b>   <b>false</b> (read-only)
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Since</b>	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](#).

```
//Add additional code
...
if (objField.isVisible) {
    ...
}
...
//Add additional code
```

## Field.isReadOnly

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns <b>true</b> if the field on the record form cannot be edited, or <b>false</b> otherwise. For textarea fields, this property can be read or written to. For all other fields, this property is read-only.
<b>Type</b>	boolean <b>true</b>   <b>false</b>
<b>Module</b>	N/currentRecord Module
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Since</b>	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](#).

```
//Add additional code
...
if (objField.isReadOnly) {
    ...
}
...
//Add additional code
```

## Field.sublistId

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the sublist ID for the specified sublist field.
<b>Type</b>	string (read-only)

<b>Module</b>	N/currentRecord Module
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Since</b>	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](#).

```
//Add additional code
...
var myId = field.sublistId;
...
//Add additional code
```

## Field.type

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the type of a body or sublist field.  For example, the value can return <code>text</code> , <code>date</code> , <code>currency</code> , <code>select</code> , <code>checkbox</code> , and other similar values. For more information on possible return values, see <a href="#">format.Type</a>  The maximum character limit for select field types is 801.
<b>Type</b>	string (read-only)
<b>Module</b>	N/currentRecord Module
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Since</b>	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](#).

```
//Add additional code
...
var type = objField.type;
...
//Add additional code
```

## currentRecord.Sublist

<b>Object Description</b>	Encapsulates a sublist on the current record.
---------------------------	---

	For a complete list of this object's methods and properties, see <a href="#">N/currentRecord Module</a> .
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	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](#).

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

## Sublist.getColumn(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a column in the sublist.
<b>Returns</b>	<a href="#">currentRecord.Column</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	The internal ID of the column field in the sublist.  See the help topic <a href="#">How do I find a field's internal ID?</a>	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 [currentRecord Module Script Samples](#).

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

## Sublist.id

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the internal ID of the sublist.
<b>Type</b>	string (read-only)
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Since</b>	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](#).

```
//Add additional code
...
var sublistid = objSublist.id;
...
//Add additional code
```

## Sublist.isChanged

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether the sublist has changed on the current record form.
<b>Type</b>	boolean <code>true</code>   <code>false</code> (read-only)
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .

<b>Since</b>	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](#).

```
//Add additional code
...
if (objSublist.isChanged) {
    ...
}
...
//Add additional code
```

## Sublist.isDisplay

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether the sublist is displayed on the current record form.
<b>Type</b>	boolean <code>true</code>   <code>false</code> (read-only)
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Since</b>	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](#).

```
//Add additional code
...
if (objSublist.isDisplay) {
    ...
}
...
//Add additional code
```

## Sublist.type

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the sublist type.
<b>Type</b>	string (read-only)

<b>Module</b>	N/currentRecord Module
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Since</b>	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](#).

```
//Add additional code
...
var sublisttype = objSublist.type;
...
//Add additional code
```

## currentRecord.get()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Retrieves a currentRecord object that represents the record active on the current page.
<b>Returns</b>	<a href="#">currentRecord.CurrentRecord</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

## Errors

Error Code	Thrown If
CANNOT_CREATE_RECORD_INSTANCE	The current record page is not scriptable or an error occurred when creating the record 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/currentRecord Module Script Sample](#).

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

```
//Add additional code
```

## currentRecord.get.promise()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Retrieves a promise for a currentRecord object that represents the record active on the current page.   <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">currentRecord.get()</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	<a href="#">Promise Object</a>
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/currentRecord Module</a>
<b>Since</b>	2016.2

### Errors

Error Code	Thrown If
CANNOT_CREATE_RECORD_INSTANCE	The current record page is not scriptable or an error occurred when creating the record instance.

### Syntax

 **Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
//Add additional code
...
var record = currentRecord.get.promise();
...
//Add additional code
```

## N/email Module

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	email.send(options)	void	Client and server scripts	Sends transactional email to an individual or group of recipients and receives bounceback notifications.
	email.send.promise(options)	void	Client and server scripts	Sends transactional email asynchronously to an individual or group of recipients and receives bounceback notifications.
	email.sendBulk(options)	void	Client and server scripts	Sends bulk email (for use when a bounceback notification is not required).
	email.sendBulk.promise(options)	void	Client and server scripts	Sends bulk email asynchronously (for use when a bounceback notification is not required).
	email.sendCampaignEvent(options)	number	Client and server scripts	Sends a single "on-demand" campaign email to a specified recipient and return a campaign response ID.
	email.sendCampaignEvent.promise(options)	number	Client and server scripts	Sends a single "on-demand" campaign email asynchronously to a specified recipient and return a campaign response ID.

## N/email Module Script Sample

The following script sample demonstrates how to use the features of the N/email module.

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to send an email with an attachment.

```
/*
 * @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,
            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
});

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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends email to an individual or group of recipients and receives bounceback notifications.  A maximum of 10 recipients (recipient + cc + bcc) is allowed.  The total message size (including attachments) must be 20MB or less. The size of each individual attachment cannot exceed 10MB.
<b>Returns</b>	void
<b>Supported Script Types</b>	Client and server scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	20 usage units

<b>Module</b>	N/email Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.author</code>	number	required	Internal ID of the email sender.  To find the internal ID of the send in the UI, go to Lists > Employees.	2015.2
<code>options.recipients</code>	number[]   string[]	required	The internal ID or email address of the recipient(s).  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
			 <b>Note:</b> Only the first recipient displays on the Communication tab (under the Recipient column). To view all recipients, click <b>View</b> to open the Message record.	
<code>options.replyTo</code>	string	optional	The email address that appears in the reply-to header.  You can use either a single external email address or a generic email address created by the Email Capture Plug-in.	2015.2
<code>options.cc</code>	number   number[]   string   string[]	optional	The internal ID or email address of the secondary recipient(s) 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
<code>options.bcc</code>	number   number[]   string   string[]	optional	The internal ID or email address of the recipient(s) 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

Parameter	Type	Required / Optional	Description	Since
options.subject	string	required	Subject of the outgoing message.	2015.2
options.body	string	required	Contents of the outgoing message.  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. To display XML as plain text, use an HTML <pre> tag around the XML.	2015.2
options.attachments	file.File	optional	Email file attachments.  You can send multiple attachments of any media type.  An individual attachment must not exceed 10MB and the total message size must be 20MB or less.  <b>Note:</b> Supported for server scripts only.	2015.2
options.RelatedRecords	Object	optional	Object that contains key/value pairs to associate (attach) the Message record with related records (i.e., transaction, activity, entity, and custom records).  See the <a href="#">RelatedRecords</a> table for more information.	2015.2
options.isInternalOnly	boolean	optional	If true, the Message record is not visible to an external Entity (for example, a customer or contact).  The default value is <b>false</b> .	2015.2

## RelatedRecords



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

Represents the NetSuite records to which an email Message record should be attached. 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 parameter (each parameter represents applicable record types).

Parameter	Type	Required / Optional	Description	Since
transactionId	number	optional	The Transaction record to attach the Message record to.  Use for transaction and opportunity record types.	2015.2
activityId	number	optional	The Activity record to attach the Message record to.	2015.2

Parameter	Type	Required / Optional	Description	Since
			Use for Case and Campaign record types.	
entityId	number	optional	The Entity record to attach the Message record to.  Use for all Entity record types (for example, customer, contact).	2015.2
customRecord	Object	optional	The custom record to attach the Message record to.  For custom records you must specify both the record ID and the record type ID.  The custom record is linked by using a nested JavaScript object.	2015.2
customRecord.id	number	optional	The instance ID for the custom record to attach the Message record to.  <b>Note:</b> If you use this parameter, <code>customRecord.recordType</code> is required.	2015.2
customRecord.recordType	string	optional	The integer ID for the custom record type to attach the Message record to. This ID is shown as part of the record's URL.  For example: <code>/custrecordentry.nl?rectype=2&amp;id=56</code> .  <b>Note:</b> If you use this parameter, <code>customRecord.id</code> is required.	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/email Module Script Sample](#).

```
//Add additional code
.

var senderId = -5;
var recipientEmail = 'notify@myCompany.com';
var timeStamp = new Date().getUTCMilliseconds();
var recipientId = 12;
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
        }
    }
});

...
//Add additional code

```

## email.send.promise(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends email asynchronously to an individual or group of recipients and receives bounceback notifications.   <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">email.send(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">email.send(options)</a>
<b>Supported Script Types</b>	Client and server scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	20 usage units
<b>Module</b>	<a href="#">N/email Module</a>
<b>Since</b>	2015.2

## email.sendBulk(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends bulk email (for use when a bounceback notification is not required).  A maximum of 10 recipients (recipient + cc + bcc) is allowed.  The total message size (including attachments) must be 20MB or less. The size of each individual attachment cannot exceed 10MB.   <b>Note:</b> This API normally uses a bulk email server to send messages. If you need to increase the successful delivery rate of an email, use <a href="#">email.send(options)</a> so that a transactional email server is used.
<b>Returns</b>	void
<b>Supported Script Types</b>	Client and server scripts

	For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 usage units
<b>Module</b>	<a href="#">N/email Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.author</code>	number	required	Internal ID of the email sender.  To find the internal ID of the send in the UI, go to Lists > Employees.	2015.2
<code>options.recipients</code>	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
			 <b>Note:</b> Only the first recipient displays on the Communication tab (under the Recipient column). To view all recipients, click <b>View</b> to open the Message record.	
<code>options.replyTo</code>	string	optional	The email address that appears in the reply-to header.  You can use either a single external email address or a generic email address created by the Email Capture Plug-in.	2015.2
<code>options.cc</code>	number   number[]   string   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
<code>options.bcc</code>	number   number[]   string   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.	2015.2

Parameter	Type	Required / Optional	Description	Since
			A maximum of 10 recipients (recipient + cc + bcc) is allowed.	
options.subject	string	required	Subject of the outgoing message.	2015.2
options.body	string	required	Contents of the outgoing message.  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. To display XML as plain text, use an HTML <pre> tag around the XML.	2015.2
options.attachments	file.File	optional	The email file attachments.  You can send multiple attachments of any media type.  An individual attachment must not exceed 10MB and the total message size must be 20MB or less.  <b>Note:</b> Supported for server scripts only.	2015.2
options.RelatedRecords	Object	optional	Object that contains key/value pairs to associate (attach) the Message record with related records (i.e., transaction, activity, entity, and custom records).  See the <a href="#">RelatedRecords</a> table for more information.	2015.2
options.isInternalOnly	boolean	optional	If true, the Message record is not visible to an external Entity (for example, a customer or contact).  The default value is <b>false</b> .	2015.2

## RelatedRecords



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

Represents the NetSuite records to which an email Message record should be attached. 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 parameter (each parameter represents applicable record types).

Parameter	Type	Required / Optional	Description	Since
transactionId	number	optional	The Transaction record to attach the Message record to.  Use for transaction and opportunity record types.	2015.2

Parameter	Type	Required / Optional	Description	Since
activityId	number	optional	The Activity record to attach the Message record to.  Use for Case and Campaign record types.	2015.2
entityId	number	optional	The Entity record to attach the Message record to.  Use for all Entity record types (for example, customer, contact).	2015.2
customRecord	Object	optional	The custom record to attach the Message record to.  For custom records you must specify both the record ID and the record type ID.  The custom record is linked by using a nested JavaScript object.	2015.2
customRecord.id	number	optional	The instance ID for the custom record to attach the Message record to.  <b>Note:</b> If you use this parameter, <code>customRecord.recordType</code> is required.	2015.2
customRecord.recordType	string	optional	The integer ID for the custom record type to attach the Message record to. This ID is shown as part of the record's URL.  For example: <code>/custrecordentry.n1?rectype=2&amp;id=56</code> .  <b>Note:</b> If you use this parameter, <code>customRecord.id</code> is required.	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/email Module Script Sample](#).

```
//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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends bulk email asynchronously (for use when a bounceback notification is not required).
	 <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">email.sendBulk(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">email.sendBulk(options)</a>
<b>Supported Script Types</b>	Client and server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 usage units
<b>Module</b>	<a href="#">N/email Module</a>
<b>Since</b>	2015.2

## email.sendCampaignEvent(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends a single “on-demand” campaign email to a specified recipient and return a campaign response ID to track the email. This is used for lead nurturing campaigns (drip marketing email).  Email (campaignemail) sublists are not supported. The campaign must use a Lead Nurturing ( <b>campaigndrip</b> ) sublist.
	 <b>Note:</b> This API normally uses a bulk email server to send messages. If you need to increase the successful delivery rate of an email, use <a href="#">email.send(options)</a> so that a transactional email server is used.
<b>Returns</b>	A campaign response ID (tracking code) as number  If the email fails to send, the value returned is -1.
<b>Supported Script Types</b>	Client and server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 usage units
<b>Module</b>	<a href="#">N/email Module</a>
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.campaignEventId	number	required	<p>The internal ID of the campaign event.</p> <p><b>Note:</b> This campaign must use a Lead Nurturing (<code>campaignndrip</code>) sublist. For more information on Lead Nurturing campaigns, see the help topic <a href="#">Lead Nurturing Campaigns</a>.</p>	2015.2
options.recipientId	number	required	<p>The internal ID of the recipient.</p> <p><b>Note:</b> The recipient's record must contain an email address.</p>	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/email Module Script Sample](#).

**Note:** The following script includes sample values for some fields. You must replace these values with values from your NetSuite account in order for the code to execute successfully.

```
// Add additional code
...
// Create a campaign record in dynamic mode so all field values can be dynamically sourced.
var campaign1 = record.create({
    type: record.Type.CAMPAIGN,
    isDynamic: true
});
campaign1.setValue({
    fieldId: 'title',
    value: 'Sample Lead Nurturing Campaign'
});
// set values on the Lead Nurturing (campaignndrip) sublist
campaign1.selectNewLine({
    sublistId: 'campaignndrip'
});
// 4 is a sample ID representing an existing marketing campaign
campaign1.setCurrentSublistValue({
    sublistId: 'campaignndrip',
    fieldId: 'template',
    value: 4
});
campaign1.setCurrentSublistValue({
    sublistId: 'campaignndrip',
    fieldId: 'title',
    value: 'Sample Lead Nurturing Event'
});
```

```

// 1 is a sample ID representing an existing subscription
campaign1.setCurrentSublistValue({
    sublistId: 'campaigndrip',
    fieldId: 'subscription',
    value: 1
});

// 2 is a sample ID representing an existing channel
campaign1.setCurrentSublistValue({
    sublistId: 'campaigndrip',
    fieldId: 'channel',
    value: 2
});

// 1 is a sample ID representing an existing promocode
campaign1.setCurrentSublistValue({
    sublistId: 'campaigndrip',
    fieldId: 'promocode',
    value: 1
});
campaign1.commitLine({
    sublistId: 'campaigndrip'
});

// Submit the record var
campaign1Key = campaign1.save();

// Load the campaign record you just created. Determine the internal ID of the campaign event to the variable
campaign2_campaigndrip_internalid_1.
var campaign2 = record.load({
    type: record.Type.CAMPAIGN,
    id : campaign1Key,
    isDynamic: true
});
var campaign2_campaigndrip_internalid_1 = campaign2.getSublistValue({
    sublistId: 'campaigndrip',
    fieldId: 'internalid',
    line: 1
});
// 142 is a sample ID representing the ID of a recipient with a valid email address
var campaignResponseId = email.sendCampaignEvent({
    campaignEventId: campaign2_campaigndrip_internalid_1,
    recipientId: 142
});
...
//Add additional code

```

## email.sendCampaignEvent.promise(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends a single “on-demand” campaign email asynchronously to a specified recipient and return a campaign response ID to track the email. This is used for lead nurturing campaigns (drip marketing email). If the email fails to send, the value returned is -1.
---------------------------	---

	<b>Note:</b> The parameters and errors thrown for this method are the same as those for <code>email.sendCampaignEvent(options)</code> . For more information on promises, see <a href="#">Promise Object</a> .
Returns	Promise Object
Synchronous Version	<code>email.sendCampaignEvent(options)</code>
Supported Script Types	Client and server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Governance	10 usage units
Module	N/email Module
Since	2015.2

## N/encode Module

**Note:** The content in this help topic pertains to SuiteScript 2.0.

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 Samples](#)

### N/encode Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<code>encode.convert(options)</code>	string	Server-side scripts	Converts a string to another type of encoding and returns the re-encoded string.
Enum	<code>encode.Encoding</code>	enum	Server-side scripts	Holds the string values for supported encoding specifications.

## N/encode Module Script Samples

The following script samples demonstrate how to use the features of the N/encode module.

### Sample 1: Convert a string to a different encoding

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following script shows how to convert a string to a different encoding.

```

/**
 * @NApiVersion 2.x
 */

require(['N/encode'], function(encode) {
    function convertStringToDifferentEncoding() {
        var stringInput = "TÃést StriÃ±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();
});

```

## encode.convert(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Converts a string to another type of encoding.
<b>Returns</b>	The re-encoded string
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/encode Module
<b>Since</b>	2015.1

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.string	string	required	The string to encode.	2015.1
options.inputEncoding	string	required	The encoding used on the input string. The default value is <code>UTF_8</code> .  Use the <code>encode.Encoding</code> to set the value.	2015.1

Parameter	Type	Required / Optional	Description	Since
options.outputEncoding	string	required	The encoding to apply to the output string. The default value is <code>UTF_8</code> .  Use the <a href="#">encode.Encoding</a> to set the value.	2015.1

## Syntax

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

```
//Add additional code
...
var hexEncodedString = encode.convert({
    string: stringInput,
    inputEncoding: encode.Encoding.UTF_8,
    outputEncoding: encode.Encoding.HEX
});
...
//Add additional code
```

## encode.Encoding

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the string values for the supported character set encoding.  This enum is used to set the value of <code>inputEncoding</code> and <code>outputEncoding</code> parameters that are members of the <a href="#">N/crypto Module</a> or <a href="#">N/encode Module</a> .   <b>Note:</b> 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.
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/encode Module</a>
<b>Since</b>	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 [encode Module Script Sample](#).

```
//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 N/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](#)

Also refer to the [N/log Module](#) for additional error logging capabilities.

## N/error Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">error.SuiteScriptError</a>	Object	Server scripts that are not user event scripts	Encapsulates a custom SuiteScript error for any server script type that is not a user event script.
	<a href="#">error.UserEventError</a>	Object	User event scripts	Encapsulates a custom SuiteScript error for a user event script.
Method	<a href="#">error.create(options)</a>	<a href="#">error.SuiteScriptError</a> or <a href="#">error.UserEventError</a>	Server scripts	Creates a new <a href="#">error.SuiteScriptError</a> or <a href="#">error.UserEventError</a> object.

## SuiteScriptError Object Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	<a href="#">SuiteScriptError.id</a>	string (read-only)	Server scripts that are not user event scripts	Error ID that is automatically generated when a new error is created.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	SuiteScriptError.message	string (read-only)	Server scripts that are not user event scripts	Error message text displayed in the Details column of the Execution Log.
	SuiteScriptError.name	string (read-only)	Server scripts that are not user event scripts	User-defined error code.
	SuiteScriptError.stack	Array of strings (read-only)	Server scripts that are not user event scripts	List of method calls that the script is executing when the error is thrown.

## UserEventError Object Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	UserEventError.eventType	string (read-only)	User event scripts	User event type ( <code>beforeLoad</code> , <code>beforeSubmit</code> , <code>afterSubmit</code> )
	UserEventError.id	string (read-only)	User event scripts	Error ID that is automatically generated when a new error is created.
	UserEventError.message	string (read-only)	User event scripts	Error message text displayed in the Details column of the Execution Log.
	UserEventError.name	string (read-only)	User event scripts	User-defined error code.
	UserEventError.recordId	string (read-only)	User event scripts	Internal ID of the submitted record that triggered the script. This property only holds a value when the error is thrown by an <code>afterSubmit</code> user event.
	UserEventError.stack	Array of strings (read-only)	User event scripts	List of method calls that the script is executing when the error is thrown.

## N/error Module Script Samples

The following script samples show two ways of creating a custom error using the N/error module.

### Sample 1: Create an error

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following samples shows how to create a custom error. If this sample code is used in a server script that is not a user event script, `errorObj` will be an `error.SuiteScriptError` object. If this sample code is used in a user event script, `errorObj` will be an `error.UserEventError` object.

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

require(['N/error'], function(error) {
    function createError() {
        var myCustomError = error.create({
            name: 'MY_ERROR_CODE',
            message: 'My custom error details',
            notifyOff: true
        });
    }

    createError();
});
```

## Sample 2: Create an error based on a condition

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following samples shows how to conditionally create and throw a custom error. If this sample code is used in a server script that is not a user event script, `errorObj` will be an `error.SuiteScriptError` object. If this sample code is used in a user event script, `errorObj` will be an `error.UserEventError` object.

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

require(['N/error'], function(error) {
    function showError() {
        var someVariable = false;

        if (!someVariable) {
            var myCustomError = error.create({
                name: 'WRONG_PARAMETER_TYPE',
                message: 'Wrong parameter type selected.',
                notifyOff: false
            });

            // This will write 'Error: WRONG_PARAMETER_TYPE Wrong parameter type selected' to the log
            log.error('Error: ' + myCustomError.name , myCustomError.message);
            throw errorObj;
        }
    }

    showError();
});
```

## error.SuiteScriptError

<b>Object Description</b>	Encapsulates a custom SuiteScript error for any server script type that is not a user event script.  Use this object in a try-catch statement to abort script execution.  Create a new custom error with the <a href="#">error.create(options)</a> method. The <a href="#">error.create(options)</a> method returns <b>error.SuiteScriptError</b> when it is called in any server script that is not a user event script.
	<p><b>Note:</b> When <a href="#">error.create(options)</a> is called in a user event script, it returns <a href="#">error.UserEventError</a>.</p>
	For a complete list of this object's methods and properties, see <a href="#">SuiteScriptError Object Members</a> .
<b>Supported Script Types</b>	All server scripts that are not user event scripts.  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/error Module</a>
<b>Since</b>	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 Sample](#).

```
//Add additional code
...
// include this code in a server script that is not a user event script to create an error.SuiteScriptError
object
var SScustom_error = error.create({
    name: 'MY_ERROR_CODE',
    message: 'my custom SuiteScriptError details',
    notifyOff: false
});
...
//Add additional code
```

## SuiteScriptError.id

<b>Property Description</b>	Error ID that is automatically generated when a new error is created.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All server scripts that are not user event scripts.  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/error Module</a>
<b>Since</b>	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 Sample.

```
//Add additional code
...
var SScustom_error = error.create({
  name: 'MY_ERROR_CODE',
  message: 'my custom SuiteScriptError details',
  notifyOff: false
});
log.debug("Error ID: " + SScustom_error.id); // id is system generated
                                              // 'Error ID: ' followed by the id will be logged
...
//Add additional code
```

## SuiteScriptError.message

<b>Property Description</b>	Error message text displayed in the Details column of the Execution Log.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All server scripts that are not user event scripts. For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/error Module</a>
<b>Since</b>	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 Sample.

```
//Add additional code
...
var SScustom_error = error.create({
  name: 'MY_ERROR_CODE',
  message: 'my custom SuiteScriptError details',
  notifyOff: false
});
log.debug("Error Message: " + SScustom_error.message); // 'Error Message: my custom SuiteScriptError details'
                                                       will be logged
...
//Add additional code
```

## SuiteScriptError.name

<b>Property Description</b>	User-defined error code.
-----------------------------	--------------------------

<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All server scripts that are not user event scripts. For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/error Module</a>
<b>Since</b>	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 Sample](#).

```
//Add additional code
...
var SScustom_error = error.create({
    name: 'MY_ERROR_CODE',
    message: 'my custom SuiteScriptError details',
    notifyOff: false
});
log.debug("Error Code: " + SScustom_error.name); // 'Error Code: MY_ERROR_CODE' will be logged
...
//Add additional code
```

## SuiteScriptError.stack

<b>Property Description</b>	List of method calls that the script is executing when the error is thrown. The most recently executed method is listed at the top of the list.
<b>Type</b>	Array of strings (read-only)
<b>Supported Script Types</b>	All server scripts that are not user event scripts. For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/error Module</a>
<b>Since</b>	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 Sample](#).

```
//Add additional code
...
var SScustom_error = error.create({
    name: 'MY_ERROR_CODE',
    message: 'my custom SuiteScriptError details',
    notifyOff: false
});
log.debug("Error Stack: " + SScustom_error.stack); // stack is set by the system
                                                // 'Error Stack: ' followed by the stack will be logged
```

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

## error.UserEventError

<b>Object Description</b>	Encapsulates a custom SuiteScript error for a user event script.  Use this object in a try-catch statement to abort script execution.  Create a new custom error with the <a href="#">error.create(options)</a> method. The <a href="#">error.create(options)</a> method returns <b>error.UserEventError</b> when it is called in a user event script.
	<p> <b>Note:</b> When <a href="#">error.create(options)</a> is called in a server script that is not a user event script, it returns <a href="#">error.SuiteScriptError</a>.</p>
	For a complete list of this object's methods and properties, see <a href="#">UserEventError Object Members</a> .
<b>Supported Script Types</b>	User event scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 User Event Script Type</a> .
<b>Module</b>	<a href="#">N/error Module</a>
<b>Since</b>	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 Sample](#).

```
//Add additional code
...
// include this code in a user event script to create an error.UserEventError object
var UEcustom_error = error.create({
    name: 'MY_ERROR_CODE',
    message: 'my custom UserEventError details',
    notifyOff: false
});
...
//Add additional code
```

## UserEventError.eventType

<b>Property Description</b>	User event type ( <code>beforeLoad</code> , <code>beforeSubmit</code> , or <code>afterSubmit</code> )
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	User event scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 User Event Script Type</a> .
<b>Module</b>	<a href="#">N/error Module</a>
<b>Since</b>	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 Sample.

```
//Add additional code
...
var UEcustom_error = error.create({
  name: 'MY_ERROR_CODE',
  message: 'my custom UserEventError details',
  notifyOff: false
});
log.debug("User Event Type: " + UEcustom_error.eventType); // eventType is set by the system
// 'User Event Type: ' followed by the eventType will
be logged
...
//Add additional code
```

## UserEventError.id

<b>Property Description</b>	Error ID that is automatically generated when a new error is created.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	User event scripts For more information, see the help topic <a href="#">SuiteScript 2.0 User Event Script Type</a> .
<b>Module</b>	N/error Module
<b>Since</b>	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 Sample.

```
//Add additional code
...
var UEcustom_error = error.create({
  name: 'MY_ERROR_CODE',
  message: 'my custom UserEventError details',
  notifyOff: false
});
log.debug("Error ID: " + UEcustom_error.id); // id is system generated
// 'Error ID: ' followed by the id will be logged
...
//Add additional code
```

## UserEventError.message

<b>Property Description</b>	Error message text displayed in the Details column of the Execution Log.
<b>Type</b>	string (read-only)

<b>Supported Script Types</b>	User event scripts For more information, see the help topic <a href="#">SuiteScript 2.0 User Event Script Type</a> .
<b>Module</b>	<a href="#">N/error Module</a>
<b>Since</b>	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 Sample](#).

```
//Add additional code
...
var UEcustom_error = error.create({
    name: 'MY_ERROR_CODE',
    message: 'my custom UserEventError details',
    notifyOff: false
});
log.debug("Error Message: " + UEcustom_error.message); // 'Error Message: my custom UserEventError details' will
be logged
...
//Add additional code
```

## UserEventError.name

<b>Property Description</b>	User-defined error code.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	User event scripts For more information, see the help topic <a href="#">SuiteScript 2.0 User Event Script Type</a> .
<b>Module</b>	<a href="#">N/error Module</a>
<b>Since</b>	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 Sample](#).

```
//Add additional code
...
var UEcustom_error = error.create({
    name: 'MY_ERROR_CODE',
    message: 'my custom UserEventError details',
    notifyOff: false
});
log.debug("Error Code: " + UEcustom_error.name); // 'Error Code: MY_ERROR_CODE' will be logged
...
//Add additional code
```

## UserEventError.recordId

<b>Property Description</b>	Internal ID of the submitted record that triggered the script. This property only holds a value when the error is thrown by an <code>afterSubmit</code> user event script.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	User event scripts For more information, see the help topic <a href="#">SuiteScript 2.0 User Event Script Type</a> .
<b>Module</b>	<a href="#">N/error Module</a>
<b>Since</b>	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 Sample](#).

```
//Add additional code
...
var UEcustom_error = error.create({
    name: 'MY_ERROR_CODE',
    message: 'my custom UserEventError details',
    notifyOff: false
});
log.debug("Submitted Record ID: " + UEcustom_error.recordId); // recordId is set by the system
                                                               // 'Submitted Record ID: ' followed by the recordId
will be logged
...
//Add additional code
```

## UserEventError.stack

<b>Property Description</b>	List of method calls that the script is executing when the error is thrown. The most recently executed method is listed at the top of the list.
<b>Type</b>	Array of strings (read-only)
<b>Supported Script Types</b>	User event scripts For more information, see the help topic <a href="#">SuiteScript 2.0 User Event Script Type</a> .
<b>Module</b>	<a href="#">N/error Module</a>
<b>Since</b>	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 Sample](#).

```
//Add additional code
...
var UEcustom_error = error.create({
```

```

    name: 'MY_ERROR_CODE',
    message: 'my custom UserEventError details',
    notifyOff: false
});
log.debug("Error Stack: " + UEcUSTOM_error.stack); // stack is set by the system
                                                // 'Error Stack: ' followed by the stack will be logged
...
//Add additional code

```

## error.create(options)

<b>Method Description</b>	Creates a new <a href="#">error.SuiteScriptError</a> or <a href="#">error.UserEventError</a> object.  Use the <a href="#">error.SuiteScriptError</a> or <a href="#">error.UserEventError</a> object in a try-catch statement to abort script execution.
<b>Returns</b>	One of the following: <ul style="list-style-type: none"> <li>■ An <a href="#">error.SuiteScriptError</a> object for server side scripts other than user event scripts.</li> <li>■ An <a href="#">error.UserEventError</a> object for user event scripts.</li> </ul>
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/error Module
<b>Since</b>	2015.2

### Parameters

<b>Note:</b>	The options parameter is a JavaScript object. The table below describes the name:value pairs that make up the object.
--------------	---

Parameter	Type	Required / Optional	Description	Since
<code>options.message</code>	string	required	Error message text displayed in the Details column of the Execution Log. Sets the value for the <a href="#">SuiteScriptError.message</a> or <a href="#">UserEventError.message</a> property.  The default value is null.	2015.2
<code>options.name</code>	string	required	User-defined error code. Sets the value for the <a href="#">SuiteScriptError.name</a> or <a href="#">UserEventError.name</a> property.	2015.2
<code>options.notifyOff</code>	boolean	optional	Sets whether email notification is suppressed. If set to false, the system emails the users identified on the applicable script record's Unhandled Errors subtab when the error is thrown.  For additional information on the Unhandled Errors subtab, see the help topic <a href="#">Creating a Script Record</a> .	2015.2

## Errors

Error Code	Message	Thrown If
MISS_MANDATORY_PARAMETER		A required argument 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/error Module Script Sample.

```
//Add additional code
...
// errorObj is a error.SuiteScriptError if this code is included in a server script that is not a user event
// script.
// errorObj is a error.UserEventError if this code is included in a user event script.

var custom_error = error.create({
    name: 'MY_ERROR_CODE',
    message: 'my custom error details',
    notifyOff: false
});
log.debug("Error Code: " + custom_error.name); // 'Error Code: MY_ERROR_CODE' will be logged
...
//Add additional code
```

## N/file Module

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

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 an iterator of custom segments of a 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 Samples](#)

### N/file Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">file.File</a>	object	Server-side scripts	Encapsulates a file within NetSuite.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	<a href="#">file.Reader</a>	object	Server-side scripts	Encapsulates a reader that you can use to perform special read operations
Method	<a href="#">file.create(options)</a>	<a href="#">file.File</a>	Server-side scripts	Creates a new <a href="#">file.File</a> .
	<a href="#">file.delete(options)</a>	void	Server-side scripts	Deletes an existing <a href="#">file.File</a> from the NetSuite File Cabinet.
	<a href="#">file.load(options)</a>	<a href="#">file.File</a>	Server-side scripts	Loads an existing <a href="#">file.File</a> from the NetSuite File Cabinet.
Enum	<a href="#">file.Encoding</a>	enum	Server-side scripts	Sets the value of the <a href="#">File.encoding</a> property.
	<a href="#">file.Type</a>	enum	Server-side scripts	Sets the value of the <a href="#">File.fileType</a> property.

## File Object Members

The following members are called on [file.File](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">File.appendLine(options)</a>	<a href="#">file.File</a> Object	Server-side scripts	Inserts a line to the end of a CSV or text file.
	<a href="#">File.getContents()</a>	string	Server-side scripts	Returns the content of a file in string format.
	<a href="#">File.lines.iterator()</a>	boolean <code>true</code>   <code>false</code>	Server-side scripts	Calls a developer-defined function for each line. Returns false when line processing stops.
	<a href="#">File.resetStream()</a>	void	Server-side scripts	Resets the file stream to its previous state.
	<a href="#">File.save()</a>	number	Server-side scripts	Saves a new or updated file to the File Cabinet.
	<a href="#">File.getReader()</a>	object	Server-side scripts	Returns reader object for read operations.
	<a href="#">File.getSegments(options)</a>	object	Server-side scripts	Returns an iterator of segments that are delimited by the specified separator.
Property	<a href="#">File.description</a>	string	Server-side scripts	Description of a file.
	<a href="#">File.encoding</a>	string	Server-side scripts	Character encoding on a file.
	<a href="#">File.fileType</a>	enum	Server-side scripts	File type of a file.
	<a href="#">File.folder</a>	number	Server-side scripts	Internal ID of the folder that houses a file within the NetSuite File Cabinet.
	<a href="#">File.id</a>	number (read-only)	Server-side scripts	Internal ID of a file in the NetSuite File Cabinet.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	File.isInactive	boolean true   false	Server-side scripts	Inactive status of a file. If set to true, the file is inactive.
	File.isOnline	boolean true   false	Server-side scripts	"Available without Login" status of a file. If set to true, users can download the file outside of a current NetSuite login session.
	File.isText	boolean (read-only)	Server-side scripts	Indicates whether a file type is text-based.
	File.name	string	Server-side scripts	Name of a file.
	File.path	string (read-only)	Server-side scripts	Relative path to a file in the NetSuite File Cabinet.
	File.size	number (read-only)	Server-side scripts	Size of a file in bytes.
	File.url	string (read-only)	Server-side scripts	URL of a file.

## Reader Object Members

The following members are called on [file.Reader](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Reader.readUntil(options)	string	Server-side scripts	Returns string from current position to the next occurrence of <code>options.tag</code> .
	Reader.readChars(options)	string	Server-side scripts	Returns the next <code>options.number</code> characters from the current position.

## N/file Module Script Samples

The following script samples demonstrate how to use the features of the N/file module.

### Sample 1: Create and save a file

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following script shows how to create and save a file to the File Cabinet. In this sample, the folder ID value is hard-coded. For the script to run in the SuiteScript Debugger, you must replace this hard-coded value with a valid folder ID from your account.

```
/**
```

```

/* @NApiVersion 2.x
*/

require(['N/file'], function(file) {
    function createAndSaveFile() {
        var fileObj = file.create({
            name: 'test.txt',
            fileType: file.Type.PLAINTEXT,
            contents: 'Hello World\nHello World'
        });
        fileObj.folder = -15;

        var id = fileObj.save();
        fileObj = file.load({
            id: id
        });
    }

    createAndSaveFile();
});

```

## Sample 2: Create and save a file using additional properties

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to create and save a file to the File Cabinet and also how to set the values of the `File.isOnline` and the `File.folder` properties. In this sample, the folder ID value is hard-coded. For the script to run in the SuiteScript Debugger, you must replace this hard-coded value with a valid folder ID from your account.

```

/**
 * @NApiVersion 2.x
*/

require(['N/file'], function(file){
    function createAndSaveFile(){
        var fileObj = file.create({
            name: 'test.txt',
            fileType: file.Type.PLAINTEXT,
            contents: 'Hello World\nHello World',
            folder : -15,
            isOnline : true
        });

        var id = fileObj.save();
        fileObj = file.load({
            id: id
        });
    }

    createAndSaveFile();
});

```

```
});
```

## Sample 3: Create a file and append lines

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample creates a CSV file, appends several lines of data, and saves the file. The script also loads the file and calculates the total of several values in the file.

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

require(['N/file', 'N/error', 'N/log'], function (file, error, log) {
    // This sample calculates the total for the
    // second column value in a CSV file.
    //
    // Each line in the CSV file has the following format:
    // date,amount
    //
    // Here is the data that the script adds to the file:
    // 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'
    });

    // Add the data
    csvFile.appendLine({
        value: '10/21/14,200.0'
    });
    csvFile.appendLine({
        value: '10/21/15,210.2'
    });
    csvFile.appendLine({
        value: '10/21/16,250.3'
    });

    // Save the file
    var csv fileId = csvFile.save();

    // Create a variable to store the calculated total
    var total = 0.0;

    // Load the file
});
```

```

var invoiceFile = file.load({
    id: csv fileId
});

// Obtain an iterator to process each line in the file
var iterator = invoiceFile.lines.iterator();

// Skip the first line, which is the CSV header line
iterator.each(function () {return false;});

// Process each line in the file
iterator.each(function (line) {
    // Update the total based on the line value
    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;
});
});

// At this point, the total is 660.5
log.debug({
    title: 'total',
    details: total
});
});
});

```

## Sample 4: Implement a basic parser

**i Note:** This script sample uses the `define` function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the `require` function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample reads and logs strings from a file using commas and new line characters as separators. This sample can be used as the starting point for a parser implementation.

```

/**
 * @NApiVersion 2.0
 * @NScriptType bankStatementParserPlugin
 */

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('\n');
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
})
}
}
});

```

## Sample 5: Read a file in segments

**i Note:** This script sample uses the **define** function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the **require** function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample reads and logs segments from a file using a set of characters as separators.

```

/**
 * @NApiVersion 2.0
 * @NScriptType bankStatementParserPlugin
 */

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

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	<p>Encapsulates a file within NetSuite.</p> <p><b>Note:</b> This object only encapsulates a file's metadata. Content is only loaded into memory (and returned as a string) when you call the <a href="#">File.getContents()</a>. Content from CSV or text files can be accessed line by line using <a href="#">File.appendLine(options)</a> or <a href="#">File.lines.iterator()</a>.</p> <p><b>Important:</b> Binary content must be base64 encoded.</p> <p>Create a new <code>file.File</code> Object (up to 10MB in size) with the <a href="#">file.create(options)</a> method. After you create a new <code>file.File</code>, you can:</p> <ul style="list-style-type: none"> <li>■ upload it to the NetSuite File Cabinet with the <a href="#">File.save()</a> method.</li> <li>■ attach it to an email or fax without saving it to the File Cabinet.</li> </ul> <p>Returns reader object <a href="#">File.getReader()</a> and iterator of segments <a href="#">File.getSegments(options)</a>. For a complete list of this object's methods and properties, see <a href="#">File Object Members</a>.</p>
<b>Supported Script Types</b>	<p>Server-side scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	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 [file Module Script Sample](#).

```

//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.getContents()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to return the content of the file.   <b>Important:</b> Content held in memory is limited to 10MB.
<b>Returns</b>	The file content as a string
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	2015.2

### Errors

Error Code	Message	Thrown If
SSS_FILE_CONTENT_SIZE_EXCEEDED	The file content you are attempting to access exceeds the maximum allowed size of 10 MB.	You attempt to return the content of a file larger than 10MB.

### Syntax

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

```
//Add additional code
...
var fileObj = file.load({
    id: 145
});
if (fileObj.size < 10485760){
    fileObj.getContents();
}
...
//Add additional code
```

## File.getReader()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to return the reader object for performing special read operations
<b>Returns</b>	<code>file.Reader</code>
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	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 [file Module Script Sample](#).

```
// 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
```

## File.getSegments(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to return the iterator of segments delimited by a separator. Separator is included in each segment. Empty separator is not allowed.
<b>Returns</b>	Iterator
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	2019.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.separator	string	required	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.	2019.1

## Errors

Error Code	Message	Thrown If
SSS_INVALID_SEGMENT_SEPARATOR	Segment separator must not be empty.	The <code>options.separator</code> argument is empty.
SSS_INVALID_ARG_TYPE	You have entered an invalid type argument: <passed type argument>	The <code>options.separator</code> argument is not a string.
SSS_MISSING_REQD_ARGUMENT	<name of missing parameter>	A required argument 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 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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to insert a line to the end of a file.
---------------------------	--

	This method can be used on text or .csv files.
	<b>Important:</b> Content held in memory is limited to 10MB. Therefore, each line must be less than 10MB.
Returns	<a href="#">file.File</a> Object
Supported Script Types	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Governance	None
Module	<a href="#">N/file Module</a>
Since	2017.1

## Parameters

<b>Note:</b>	The options parameter is a JavaScript object. The table below describes the name:value pairs that make up the object.
--------------	---

Parameter	Type	Required / Optional	Description	Since
options.value	string	required	Object containing a string to insert at the end of the file.	2017.1

## Errors

Error Code	Message	Thrown If
SSS_FILE_CONTENT_SIZE_EXCEEDED	The content you are attempting to access exceeds the maximum allowed size of 10 MB.	You attempt to return the content of a line larger than 10MB.
YOU_CANNOT_WRITE_TO_A_FILE_AFTER_YOU_BEGAN_READING_FROM_IT		You call <a href="#">File.appendLine(options)</a> after calling <a href="#">File.lines.iterator()</a> . To avoid receiving the error, call <a href="#">File.resetStream()</a> or save the file.

## Syntax

<b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">file Module Script Sample</a> .
---

```
//Add additional code
...
var fileObj = file.load({
    id: 145
});
fileObj.appendLine({
    value: 'hello world'
});
...
//Add additional code
```

## File.lines.iterator()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>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.</p> <p>Return <code>false</code> to stop the loop. Return true to continue the loop. By default, false is returned when the end of the file is reached.</p> <p>This method can be used on text or .csv files.</p>
	<p><b>Important:</b> Content held in memory is limited to 10MB. Therefore, each line must be less than 10MB.</p>
<b>Returns</b>	Boolean <code>true</code>   <code>false</code>
<b>Supported Script Types</b>	<p>Server-side scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	2017.1

### Parameters

Parameter	Type	Required / Optional	Description	Since
lineContext	iterator	required	Iterator which provides the next line of text from the text file to the iterator function.	2017.1

### Errors

Error Code	Message	Thrown If
SSS_FILE_CONTENT_SIZE_EXCEEDED	The content you are attempting to access exceeds the maximum allowed size of 10 MB.	You attempt to return the content of a line larger than 10MB.
YOU_CANNOT_READ_FROM_A_FILE_AFTER_YOU_BEGAN_WRITING_TO_IT		You call <code>File.lines.iterator()</code> after calling <code>File.appendLine(options)</code> . Call <code>File.resetStream()</code> or save the file.

### Syntax



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

```
//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()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to reset the file contents. Serves as an undo action on any unsaved content written with <a href="#">File.appendLine(options)</a> or <a href="#">File.lines.iterator()</a> .  Use this method to reset the reading and writing streams that may have been opened by <a href="#">File.appendLine(options)</a> or <a href="#">File.lines.iterator()</a> .  The line pointer (or read iterator) is also set to its previous state.  This method can be used on text or .csv files.
<b>Returns</b>	Void
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	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 [file Module Script Sample](#).

```

//Add additional code
...
var afile = file.create({
    name: 'tmp3.txt',
    fileType: 'PLAINTEXT',

```

```

        contents:'one line'
    });
afile.appendLine({
    value:'line two'
});
afile.resetStream();
afile.lines(function f(){});
...
//Add additional code

```

## File.save()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Method used to:</p> <ul style="list-style-type: none"> <li>■ Upload a new file to the NetSuite File Cabinet.</li> <li>■ Save an updated file to the NetSuite File Cabinet.</li> </ul> <p><b>Note:</b> The <code>File.save()</code> method streams files of any size, provided that the file to save or upload meets File Cabinet limits.</p> <p><b>Important:</b> If you want to save the file to the NetSuite File Cabinet, you must set a NetSuite File Cabinet folder with the <code>File.folder</code> property. You must do this before you call <code>File.save()</code>.</p>
<b>Returns</b>	The internal ID of the file as a number.
<b>Supported Script Types</b>	<p>Server-side scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Governance</b>	20 usage units
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
INVALID_KEY_OR_REF	Invalid folder reference key <passed folder ID>.	The <code>File.folder</code> property is set to an invalid folder ID.
SSS_MISSING_REQD_ARGUMENT	Please enter value(s) for: Folder	The <code>File.folder</code> property is not set before <code>save()</code> is called.

## Syntax



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

```
//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.description

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The description of a file. In the UI, the value of <b>description</b> displays in the <b>Description</b> field on the file record.
<b>Type</b>	string
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	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 [file Module Script Sample](#).

```

//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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The character encoding on a file. Value is set with the <a href="#">file.Encoding</a> enum.
<b>Type</b>	string
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Module</b>	N/file Module
<b>Since</b>	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 file Module Script Sample.

```
//Add additional code
...
var fileObj = file.create({
    name      : 'test.txt',
    fileType: file.Type.PLAINTEXT,
    contents: 'Hello World\nHello World'
});
fileObj.encoding = file.Encoding.MAC_ROMAN;
fileObj.folder = 30;
var fileId = fileObj.save();
...

//Add additional code
```

## File.fileType

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The file type of a file.  This property is read-only. You must set the file type by passing in a <a href="#">file.Type</a> enum value to <a href="#">file.create(options)</a> .
<b>Type</b>	enum
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/file Module
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property after it is set with <a href="#">file.create(options)</a> .

## Syntax

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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal ID of a file's folder within the NetSuite File Cabinet.  Before you upload a file to the NetSuite File Cabinet with <a href="#">File.save()</a> , you must set its File Cabinet folder with the <b>folder</b> property.
<b>Type</b>	number   string
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	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 [file Module Script Sample](#).

```

//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.id

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal ID of the file within the NetSuite File Cabinet. This value is automatically generated by NetSuite.
-----------------------------	--

	This property is read-only.
<b>Type</b>	number
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax

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

```
//Add additional code
...
var fileObj = file.load({
    id: 'Images/myImageFile.jpg'
});
log.debug({
    details: "File ID: " + fileObj.id
});
...
//Add additional code
```

## File.isInactive

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The inactive status of a file. If set to true, the file is inactive.  The default value is <b>false</b> .  When a file is inactive, it does not display in the UI unless you select <b>Show Inactives</b> on the File Cabinet page.
<b>Type</b>	boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	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 file Module Script Sample.

```
//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.isOnLine

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>The <b>Available without Login</b> status of a file. If set to <b>true</b>, users can download the file outside of a current NetSuite login session.</p> <p>The default value is <b>false</b>.</p> <p><b>Important:</b> This property holds the value of the <b>Available without Login</b> setting found on the file record. It does not reflect the value of the <b>Available Without Login</b> setting found on the Suitelet script deployment record.</p> <p>The <b>Available without Login</b> 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.</p>
<b>Type</b>	boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	<p>Server-side scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	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 file Module Script Sample.

```
//Add additional code
...
var fileObj = file.load({
    id: 'Images/myImageFile.jpg'
});
fileObj.isOnLine = true;
var fileId = fileObj.save();
```

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

## File.isText



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether a file type is text-based. This property is read-only.
<b>Type</b>	boolean <code>true</code>   <code>false</code>
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	2015.2

### Errors

Error Code	Message	Thrown If
<code>READ_ONLY_PROPERTY</code>		You attempt to edit this property.

### Syntax



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

```
//Add additional code
...
var fileObj = file.load({
    id: 145
});
if (fileObj.isText === true){
    ...
}
...
//Add additional code
```

## File.name



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The name of a file.
<b>Type</b>	string
<b>Supported Script Types</b>	Server-side scripts

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	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 file Module Script Sample.

```
//Add additional code
...
var fileObj = file.load({
    id: 'Images/myImageFile.jpg'
});
fileObj.name = 'myOldImageFile.jpg';
var fileId = fileObj.save();
...
//Add additional code
```

## File.path

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The relative path to a file in the NetSuite File Cabinet.   <b>Note:</b> If the folder is not set with the <a href="#">file.create(options)</a> method, this property holds the file name until the <a href="#">File.folder</a> property is defined.
<b>Type</b>	string
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax

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

```
//Add additional code
```

```

...
var fileObj = file.load({
    id: 145
});
log.debug({
    details: "File Path: " + fileObj.path
});
...
//Add additional code

```

## File.size



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The size of a file in bytes.  This property is read-only.
<b>Type</b>	number
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	2015.2

### Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

### Syntax



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

```

//Add additional code
...
var fileObj = file.load({
    id: 'Images/myImageFile.jpg'
});
log.debug({
    details: "File Size: " + fileObj.size
});
...
//Add additional code

```

## File.url



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The URL of a file. This property is read-only.
<b>Type</b>	string
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	2015.2

### Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

### Syntax



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

```
//Add additional code ...
var fileObj = file.load({
    id: 'Images/myImageFile.jpg'
});
log.debug({
    details: "File URL: " + fileObj.url
});
...
//Add additional code
```

## file.create(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to create a new file in the NetSuite File Cabinet.
	<b>Important:</b> Content held in memory is limited to 10MB.
<b>Returns</b>	<a href="#">file.File</a>
<b>Supported Script Types</b>	Server-side scripts

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/file Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	<ul style="list-style-type: none"> <li>■ The file name and extension.</li> <li>■ Sets the value for the <a href="#">File.name</a> property.</li> </ul>	2015.2
options.fileType	enum	required	<ul style="list-style-type: none"> <li>■ The file type.</li> <li>■ Sets the value for the <a href="#">File.fileType</a> property. This property is read-only and cannot be changed after the file is created.</li> <li>■ Use the <a href="#">file.Type</a> enum to set the value.</li> </ul>	2015.2
options.contents	string	optional	<ul style="list-style-type: none"> <li>■ The file content.</li> <li>■ File content is lazy loaded; there is no property for it.</li> <li>■ If the file type is binary (for example, PDF), the file content must be base64 encoded.</li> </ul>	2015.2
options.description	string	optional	<ul style="list-style-type: none"> <li>■ The file description. In the UI, the value of description displays the <b>Description</b> field on the file record.</li> <li>■ Sets the value for the <a href="#">File.description</a> property.</li> </ul>	2016.2
options.folder	number	optional	<ul style="list-style-type: none"> <li>■ 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 <a href="#">File.save()</a>.</li> <li>■ Sets the value for the <a href="#">File.folder</a> property.</li> </ul>	2016.1
options.encoding	string	optional	<ul style="list-style-type: none"> <li>■ The character encoding on a file.</li> <li>■ Sets the value for the <a href="#">File.encoding</a> property.</li> <li>■ Use the <a href="#">file.Encoding</a> enum to set the value.</li> </ul>	2016.2
options.isInactive	boolean <code>false</code>   <code>true</code>	optional	<ul style="list-style-type: none"> <li>■ 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 <b>Show Inactives</b> on the File Cabinet page.</li> <li>■ Sets the value for the <a href="#">File.isInactive</a> property.</li> </ul>	2016.2
options.isOnline	boolean <code>false</code>   <code>true</code>	optional	<ul style="list-style-type: none"> <li>■ The <b>Available without Login</b> status of a file. If set to true, users can download the file</li> </ul>	2016.2

Parameter	Type	Required / Optional	Description	Since
			outside of a current netSuite login session. The default value is false. <ul style="list-style-type: none"> <li>■ Sets the value for the <a href="#">File.isOnline</a> property.</li> </ul>	

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	<name of missing parameter>	A required argument is not passed.
SSS_INVALID_TYPE_ARG	You have entered an invalid type argument: <passed type argument>	The argument for <a href="#">File.fileType</a> is invalid.
SSS_FILE_CONTENT_SIZE_EXCEEDED	The file you are trying to create exceeds the maximum allowed file size of 10.0 MB.	You attempt to create a file larger than 10MB.

## Syntax

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

```
//Add additional code
...
var fileObj = file.create({
    name: 'test.txt',
    fileType: file.Type.PLAINTEXT,
    contents: 'Hello World\nHello World',
    description: 'This is a plain text file.',
    encoding: file.Encoding.UTF8,
    folder: 30,
    isOnline: true
});
...
//Add additional code
```

## file.delete(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to delete an existing file from the NetSuite File Cabinet.
<b>Returns</b>	The internal ID of the deleted file
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	20 usage units
<b>Module</b>	<a href="#">N/file Module</a>

<b>Since</b>	2015.2
--------------	--------

## Parameters

<b>i</b> <b>Note:</b> The options parameter is a JavaScript object.
---

Parameter	Type	Required / Optional	Description	Since
options.id	number   string	required	<ul style="list-style-type: none"> <li>■ Internal ID of the file.</li> <li>■ To find the internal ID of the file in the UI, click Documents &gt; Files &gt; File Cabinet.</li> </ul>	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	<name of missing parameter>	A required argument is not passed.

## Syntax

<b>!</b> <b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see file Module Script Sample.
---

```
//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();

file.delete({
    id: fileId
});
...
//Add additional code
```

## file.load(options)

<b>i</b> <b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.
---

<b>Method Description</b>	Loads an existing file from the NetSuite File Cabinet.
<b>Returns</b>	<code>file.File</code>
<b>Supported Script Types</b>	<p>Server-side scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>

<b>Governance</b>	10 usage units
<b>Module</b>	N/file Module
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.id	number   string	required	<p>The identifier of a file in the File Cabinet.</p> <p>To specify a file in the File Cabinet, you can pass one of the following as the value of this parameter:</p> <ul style="list-style-type: none"> <li>■ The internal ID of the file as a number or string</li> <li>■ The absolute file path to the file in the File Cabinet (for example, 'Images/myImageFile.jpg')</li> <li>■ The relative file path to the file in the File Cabinet (for example, './Images/myImageFile.jpg' to specify a file path relative to the current folder of your script, or '../Images/myImageFile.jpg' to specify a file path relative to the parent folder of your script)</li> </ul> <p>To find the internal ID of the file in the UI, select Documents &gt; Files &gt; File Cabinet.</p>	2015.2

## Errors

Error Code	Message	Thrown If
INSUFFICIENT_PERMISSION	You do not have access to the media item you selected.	Internal ID passed is invalid.
RCRD_DSNT_EXIST	That record does not exist. path: {path}	Relative file path passed is invalid.
SSS_MISSING_REQD_ARGUMENT	<name of missing parameter>	A required argument is not passed.

## Syntax

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

```
// 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	<p>Enumeration that holds the string values for supported character encoding. This enum is used to set the value of the <a href="#">File.encoding</a> property.</p> <p><b>Note:</b> 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>
<b>Supported Script Types</b>	<p>Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	2015.2

### Values

Value	Character Set
UTF_8	Unicode
WINDOWS_1252	Western
ISO_8859_1	Western
GB18030	Chinese Simplified
SHIFT_JIS	Japanese
MAC_ROMAN	Western
GB2312	Chinese Simplified
BIG5	Chinese Traditional

### Syntax



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

```
//Add additional code
...
var fileObj = file.create({
    name: 'test.txt',
    fileType: file.Type.PLAINTEXT,
    contents: 'Hello World\nHello World'
});
fileObj.encoding = file.Encoding.MAC_ROMAN;
fileObj.folder = 30;
var fileId = fileObj.save();
...
```

```
//Add additional code
```

## file.Type



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	<p>Enumeration that holds the string values for supported file types. This enum is used to set the value of the <a href="#">File.fileType</a> property.</p>
	<p>Note that the <a href="#">File.fileType</a> property is read only. Its value must be set with <a href="#">file.create(options)</a>. See file Module Code Sample for an example.</p>
	<p><b>Note:</b> 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>
<b>Supported Script Types</b>	<p>Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	2015.2

### Values

<ul style="list-style-type: none"> <li>■ APPCACHE</li> <li>■ AUTOCAD</li> <li>■ BMPIMAGE</li> <li>■ CERTIFICATE</li> <li>■ CONFIG</li> <li>■ CSV</li> <li>■ EXCEL</li> <li>■ FLASH</li> <li>■ FREEMARKER</li> <li>■ GIFIMAGE</li> <li>■ GZIP</li> <li>■ HTMLDOC</li> <li>■ ICON</li> <li>■ JAVASCRIPT</li> <li>■ JPGIMAGE</li> </ul>	<ul style="list-style-type: none"> <li>■ JSON</li> <li>■ MESSAGERFC</li> <li>■ MP3</li> <li>■ MPEGMOVIE</li> <li>■ MSPROJECT</li> <li>■ PDF</li> <li>■ PJPGIMAGE</li> <li>■ PLAINTEXT</li> <li>■ PNGIMAGE</li> <li>■ POSTSCRIPT</li> <li>■ POWERPOINT</li> <li>■ QUICKTIME</li> </ul>	<ul style="list-style-type: none"> <li>■ RTF</li> <li>■ SCSS</li> <li>■ SMS</li> <li>■ STYLESHEET</li> <li>■ SVG</li> <li>■ TAR</li> <li>■ TIFFIMAGE</li> <li>■ VISIO</li> <li>■ WEBAPPPAGE</li> <li>■ WEBAPPSCRIPT</li> <li>■ WORD</li> <li>■ XMLDOC</li> <li>■ XSD</li> <li>■ ZIP</li> </ul>
--	---	--

### Syntax



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

```
//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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Use for special read operations. Reads from a file until a specified delimiter is reached.  Reads an arbitrary number of characters from a file.
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/file Module</a>
<b>Methods and Properties</b>	<a href="#">Reader Object Members</a>
<b>Since</b>	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 [file Module Script Sample](#).

```

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);

```

## Reader.readUntil(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns string from current position to the next occurrence of <code>options.tag</code> .  Returns the rest of the string if tag is not found.  Returns null if reading is already finished.  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.
<b>Returns</b>	string
<b>Supported Script Types</b>	Server-side scripts

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/file Module
<b>Since</b>	2019.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.tag	string	required	String containing a tag	2019.1

## Errors

Error Code	Message	Thrown If
SSS_TAG_CANNOT_BE_EMPTY	Tag cannot be empty.	The <code>options.tag</code> argument is empty.
SSS_INVALID_ARG_TYPE	You have entered an invalid type argument: <passed type argument>	The <code>options.tag</code> argument is not a string.
SSS_MISSING_REQD_ARGUMENT	<name of missing parameter>	A required argument 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 [file Module Script Sample](#).

```
// 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)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the next <code>options.number</code> 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.
<b>Returns</b>	string

<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/file Module</a>
<b>Since</b>	2019.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.number</code>	number	required	The number of characters to read.	2019.1

## Errors

Error Code	Message	Thrown If
<code>SSS_INVALID_READ_SIZE</code>	Read size must be positive.	The <code>options.number</code> argument is not greater than zero.
<code>SSS_INVALID_ARG_TYPE</code>	You have entered an invalid type argument: <passed type argument>	The <code>options.number</code> argument is not a number.
<code>SSS_MISSING_REQD_ARGUMENT</code>	<name of missing parameter>	A required argument 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 [file Module Script Sample](#).

```
// 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
```

## N/format Module

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">format.format(options)</a>	string   Date	Client and server-side scripts	Takes a raw value and returns a formatted value.  <div style="border: 1px solid #ccc; padding: 5px; background-color: #f9f9f9;"> <span style="color: #0070C0; font-size: 1.5em; margin-right: 5px;">i</span> <b>Note:</b> This method is overloaded when you format a <code>datetime</code> or <code>datetimetz</code> value.         </div>
	<a href="#">format.parse(options)</a>	Date   string   number	Client and server-side scripts	Takes a formatted value and returns a raw value.  <div style="border: 1px solid #ccc; padding: 5px; background-color: #f9f9f9;"> <span style="color: #0070C0; font-size: 1.5em; margin-right: 5px;">i</span> <b>Note:</b> This method is overloaded when you format a <code>datetime</code> or <code>datetimetz</code> value.         </div>
Enum	<a href="#">format.Type</a>	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 <code>options.type</code> parameter.
	<a href="#">format.Timezone</a>	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 <code>options.timezone</code> 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](#).

i **Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample parses a string (formatted according to the user preference) to a raw Date Object, and then parses it back to the formatted string. This sample uses `format.parse(options)` and `format.format(options)`.

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

require(['N/format'],
```

```

function(format) {
    function parseAndFormatDateString() {
        // Assume Date format is MM/DD/YYYY
        var initialFormattedDateString = "07/28/2015";
        var parsedDateStringAsRawDateObject = format.parse({
            value: initialFormattedDateString,
            type: format.Type.DATE
        });
        var formattedDateString = format.format({
            value: parsedDateStringAsRawDateObject,
            type: format.Type.DATE
        });
    }
    parseAndFormatDateString();
    // "07/28/2015"
}
);

```

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample parses a string (formatted according to the user preference) to a raw number value, using `format.parse(options)`.

```

/**
 * @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
    });

```

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample formats a raw number value (formatted according to the user preference) to a string, using `format.format(options)`.

```

/**
 * @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
});
```

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample formats the time of day to a string, using `format.format(options)`.

```
/**
 * @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)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Formats a value from the raw value to its appropriate preference format.  <b>Note:</b> This method is overloaded when you format a <code>datetime</code> or <code>datetimetz</code> value.
<b>Returns</b>	If a <code>datetime</code> or <code>datetimetz</code> value is specified, the string in date format is returned in the user's local app time zone.  <b>Note:</b> If an invalid value is given, the original value passed to <code>options.value</code> is returned.  <b>Note:</b> 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.
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>
<b>Governance</b>	None

<b>Module</b>	N/format Module
<b>Since</b>	2015.2

## Parameters

This method is overloaded when you format a `datetime` or `datetimetz` value.

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

Parameter	Type	Required / Optional	Description	Since
<code>options.value</code>	Date   string   number	required	The input data to format.	2015.2
<code>options.type</code>	string	required	The field type (for example, DATE, CURRENCY, INTEGER).  Set using the <code>format.Type</code> enum.	2015.2

The table below applies to datetime and datetimetz values only.

Parameter	Type	Required / Optional	Description	Since
<code>options.value</code>	Date	required	The Date Object being converted into a string	2015.2
<code>options.type</code>	string	required	The field type (either DATETIME or DATETIMETZ).  Set using the <code>format.Type</code> enum.	2015.2
<code>options.timezone</code>	enum   number	optional	The time zone specified for the returned string. Set using the <code>format.Timezone</code> 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.	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/format Module Script Samples](#).

```
// 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)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <b>datetime</b> or <b>datetimetz</b> value, use this method to convert a Date Object into a string based on the specified timezone.  <b>Note:</b> This method is overloaded when you format a <b>datetime</b> or <b>datetimetz</b> value.
<b>Returns</b>	Datetime or datetimetz values are returned as a Date Object.  <b>Note:</b> If the value given is not valid or parsable, the original value passed to options.value is returned.
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>
<b>Governance</b>	None
<b>Module</b>	N/format Module
<b>Since</b>	2015.2

### Parameters

This method is overloaded when you format a **datetime** or **datetimetz** value.

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

Parameter	Type	Required / Optional	Description	Since
options.value	string	required	The input data to parse.	2015.2
options.type	string	required	The field type (for example, DATE, CURRENCY, INTEGER).  Set using the <a href="#">format.Type</a> enum.	2015.2

The table below applies to datetime and datetimeTZ values only.

Parameter	Type	Required / Optional	Description	Since
options.value	string	required	The string that contains the date and time information in the specified timezone.	2015.2

Parameter	Type	Required / Optional	Description	Since
<code>options.type</code>	string	required	The field type (either <code>DATETIME</code> or <code>DATETIMETZ</code> ). Set using the <a href="#">format.Type</a> enum.	2015.2
<code>options.timezone</code>	enum	optional	The time zone represented by the <code>options.value</code> string. Set using the <a href="#">format.Timezone</a> enum.  If a time zone is not specified, the time zone is based on user preference.  If the time zone is invalid, the time zone is set to GMT.	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/format Module Script Samples](#).

```
// 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for the supported field types. This enum is used to set the value of the <code>options.type</code> parameter when calling <code>format.format(options)</code> or <code>format.parse(options)</code> .
	<p><b>Note:</b> 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>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>
<b>Module</b>	<a href="#">N/format Module</a>

Since	2015.2
-------	--------

## Values

■ ADDRESS	■ EMAIL	■ POSCURRENCY
■ CCEXDATE	■ EMAILS	■ POSFLOAT
■ CCNUMBER	■ FLOAT	■ POSINTEGER
■ CCVALIDFROM	■ FULLPHONE	■ QUOTEDFUNCTION
■ CHECKBOX	■ FUNCTION	■ RADIO
■ COLOR	■ FURIGANA	■ RATE
■ CURRENCY	■ IDENTIFIER	■ RATEHIGHPRECISION
■ CURRENCY2	■ IDENTIFIERANYCASE	■ SELECT
■ DATE	■ INTEGER	■ TEXT
■ DATETIME	■ MMYYDATE	■ TEXTAREA
■ DATETIMETZ	■ NONNEGURRENCY	■ TIME
■ DOCUMENT	■ NONNEGFLOAT	■ TIMEOFDAY
■ DYNAMICPRECISION	■ PACKAGE	■ TIMETRACK
	■ PERCENT	■ URL
	■ PHONE	

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 any of the following field types is set to hidden, the object returned is text.
  - Checkbox
  - Radio
  - Select
  - Textarea
- DYNAMICPRECISION is controlled by NetSuite and can differ from field to field.

## 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](#).

```
// 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

```

```

require(['N/format'],
function(format) {

    function parseAndFormatDateString()
    {
        var rawDateString = "07/28/2015";
        var parsedDate= format.parse({
            value: rawDateString,
            type: format.Type.DATE
        });
        console.log(parsedDate);
    }

    parseAndFormatDateString();
}
);

```

## format.Timezone

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	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> .
	 <b>Note:</b> 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.
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>
<b>Module</b>	N/format Module
<b>Since</b>	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.

Key	Olson Value	Description
1	ETC_GMT_PLUS_12: 'Etc/GMT+12'	(GMT-12:00) International Date Line West
2	PACIFIC_SAMOA: 'Pacific/Samoa'	(GMT-11:00) Midway Island, Samoa
3	PACIFIC_HONOLULU: 'Pacific/Honolulu'	(GMT-10:00) Hawaii
4	AMERICA_ANCHORAGE: 'America/Anchorage'	(GMT-09:00) Alaska
5	AMERICA_LOS_ANGELES: 'America/Los_Angeles'	(GMT-08:00) Pacific Time (US & Canada)
6	AMERICA_TIJUANA: 'America/Tijuana'	(GMT-08:00) Tijuana, Baja California
7	AMERICA_DENVER: 'America/Denver'	(GMT-07:00) Mountain Time (US & Canada)
8	AMERICA_PHOENIX: 'America/Phoenix'	(GMT-07:00) Arizona
9	AMERICA_CHIHUAHUA: 'America/Chihuahua'	(GMT-07:00) Chihuahua, La Paz, Mazatlan - New
10	AMERICA_CHICAGO: 'America/Chicago'	(GMT-06:00) Central Time (US & Canada)
11	AMERICA_REGINA: 'America/Regina'	(GMT-06:00) Saskatchewan
12	AMERICA_GUATEMALA: 'America/Guatemala'	(GMT-06:00) Central America
13	AMERICA_MEXICO_CITY: 'America/Mexico_City'	(GMT-06:00) Guadalajara, Mexico City, Monterrey - Old
14	AMERICA_NEW_YORK: 'America/New_York'	(GMT-05:00) Eastern Time (US & Canada)
15	US_EAST_INDIANA: 'US/East-Indiana'	(GMT-05:00) Indiana (East)
16	AMERICA_BOGOTA: 'America/Bogota'	(GMT-05:00) Bogota, Lima, Quito
17	AMERICA_CARACAS: 'America/Caracas'	(GMT-04:30) Caracas
18	AMERICA_HALIFAX: 'America/Halifax'	(GMT-04:00) Atlantic Time (Canada)
19	AMERICA_LA_PAZ: 'America/La_Paz'	(GMT-04:00) Georgetown, La Paz, San Juan
20	AMERICA_MANAUS: 'America/Manaus'	(GMT-04:00) Manaus
21	AMERICA_SANTIAGO: 'America/Santiago'	(GMT-04:00) Santiago
22	AMERICA_ST_JOHNS: 'America/St_Johns'	(GMT-03:30) Newfoundland
23	AMERICA_SAO_PAULO: 'America/Sao_Paulo'	(GMT-03:00) Brasilia
24	AMERICA_BUENOS_AIRES: 'America/Buenos_Aires'	(GMT-03:00) Buenos Aires
25	ETC_GMT_PLUS_3: 'Etc/GMT+3'	(GMT-03:00) Cayenne
26	AMERICA_GODTHAB: 'America/Godthab'	(GMT-03:00) Greenland
27	AMERICA_MONTEVIDEO: 'America/Montevideo'	(GMT-03:00) Montevideo
28	AMERICA_NORONHA: 'America/Noronha'	(GMT-02:00) Mid-Atlantic
29	ETC_GMT_PLUS_1: 'Etc/GMT+1'	(GMT-01:00) Cape Verde Is.

Key	Olson Value	Description
30	ATLANTIC_AZORES: 'Atlantic/Azores'	(GMT-01:00) Azores
31	EUROPE_LONDON: 'Europe/London', GMT: 'GMT'	(GMT) Greenwich Mean Time : Dublin, Edinburgh, Lisbon, London
32	GMT: 'GMT'	(GMT) Casablanca
33	ATLANTIC_REYKJAVIK: 'Atlantic/Reykjavik'	(GMT) Monrovia, Reykjavik
34	EUROPE_WARSAW: 'Europe/Warsaw'	(GMT+01:00) Sarajevo, Skopje, Warsaw, Zagreb
35	EUROPE_PARIS: 'Europe/Paris'	(GMT+01:00) Brussels, Copenhagen, Madrid, Paris
36	ETC_GMT_MINUS_1: 'Etc/GMT-1'	(GMT+01:00) West Central Africa
37	EUROPE_AMSTERDAM: 'Europe/Amsterdam'	(GMT+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
38	EUROPE_BUDAPEST: 'Europe/Budapest'	(GMT+01:00) Belgrade, Bratislava, Budapest, Ljubljana, Prague
39	AFRICA_CAIRO: 'Africa/Cairo'	(GMT+02:00) Cairo
40	EUROPE_ISTANBUL: 'Europe/Istanbul'	(GMT+02:00) Athens, Bucharest, Istanbul
41	ASIA_JERUSALEM: 'Asia/Jerusalem'	(GMT+02:00) Jerusalem
42	ASIA_AMMAN: 'Asia/Amman'	(GMT+02:00) Amman
43	ASIA_BEIRUT: 'Asia/Beirut'	(GMT+02:00) Beirut
44	AFRICA_JOHANNESBURG: 'Africa/Johannesburg'	(GMT+02:00) Harare, Pretoria
45	EUROPE_KIEV: 'Europe/Kiev'	(GMT+02:00) Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius
46	EUROPE_MINSK: 'Europe/Minsk'	(GMT+02:00) Minsk
47	AFRICA_WINDHOEK: 'Africa/Windhoek'	(GMT+02:00) Windhoek
48	ASIA_RIYADH: 'Asia/Riyadh'	(GMT+03:00) Kuwait, Riyadh
49	EUROPE_MOSCOW: 'Europe/Moscow'	(GMT+03:00) Moscow, St. Petersburg, Volgograd
50	ASIA_BAGHDAD: 'Asia/Baghdad'	(GMT+03:00) Baghdad
51	AFRICA_NAIROBI: 'Africa/Nairobi'	(GMT+03:00) Nairobi
52	ASIA_TEHRAN: 'Asia/Tehran'	(GMT+03:30) Tehran
53	ASIA_MUSCAT: 'Asia/Muscat'	(GMT+04:00) Abu Dhabi, Muscat
54	ASIA_BAKU: 'Asia/Baku'	(GMT+04:00) Baku
55	ASIA_YEREVAN: 'Asia/Yerevan'	(GMT+04:00) Caucasus Standard Time
56	ETC_GMT_MINUS_3: 'Etc/GMT-3'	(GMT+04:00) Tbilisi
57	ASIA_KABUL: 'Asia/Kabul'	(GMT+04:30) Kabul
58	ASIA_KARACHI: 'Asia/Karachi'	(GMT+05:00) Islamabad, Karachi
59	ASIA_YEKATERINBURG: 'Asia/Yekaterinburg'	(GMT+05:00) Ekaterinburg

Key	Olson Value	Description
60	ASIA_TASHKENT: 'Asia/Tashkent'	(GMT+05:00) Tashkent
61	ASIA_CALCUTTA: 'Asia/Calcutta'	(GMT+05:30) Chennai, Kolkata, Mumbai, New Delhi
62	ASIA_KATMANDU: 'Asia/Katmandu'	(GMT+05:45) Kathmandu
63	ASIA_ALMATY: 'Asia/Almaty'	(GMT+06:00) Novosibirsk
64	ASIA_DHAKA: 'Asia/Dhaka'	(GMT+06:00) Astana, Dhaka
65	ASIA_RANGOON: 'Asia/Rangoon'	(GMT+06:30) Yangon (Rangoon)
66	ASIA_BANGKOK: 'Asia/Bangkok'	(GMT+07:00) Bangkok, Hanoi, Jakarta
67	ASIA_KRASNOYARSK: 'Asia/Krasnoyarsk'	(GMT+07:00) Krasnoyarsk
68	ASIA_HONG_KONG: 'Asia/Hong_Kong'	(GMT+08:00) Beijing, Chongqing, Hong Kong, Urumqi
69	ASIA_KUALA_LUMPUR: 'Asia/Kuala_Lumpur'	(GMT+08:00) Kuala Lumpur, Singapore
70	ASIA_TAIPEI: 'Asia/Taipei'	(GMT+08:00) Taipei
71	AUSTRALIA_PERTH: 'Australia/Perth'	(GMT+08:00) Perth
72	ASIA_IRKUTSK: 'Asia/Irkutsk'	(GMT+08:00) Irkutsk
73	ASIA_MANILA: 'Asia/Manila'	(GMT+08:00) Manila
74	ASIA_SEOUL: 'Asia/Seoul'	(GMT+09:00) Seoul
75	ASIA_TOKYO: 'Asia/Tokyo'	(GMT+09:00) Osaka, Sapporo, Tokyo
76	ASIA_YAKUTSK: 'Asia/Yakutsk'	(GMT+09:00) Yakutsk
77	AUSTRALIA_DARWIN: 'Australia/Darwin'	(GMT+09:30) Darwin
78	AUSTRALIA_ADELAIDE: 'Australia/Adelaide'	(GMT+09:30) Adelaide
79	AUSTRALIA_SYDNEY: 'Australia/Sydney'	(GMT+10:00) Canberra, Melbourne, Sydney
80	AUSTRALIA_BRISBANE: 'Australia/Brisbane'	(GMT+10:00) Brisbane
81	AUSTRALIA_HOBART: 'Australia/Hobart'	(GMT+10:00) Hobart
82	PACIFIC_GUAM: 'Pacific/Guam'	(GMT+10:00) Guam, Port Moresby
83	ASIA_VLADIVOSTOK: 'Asia/Vladivostok'	(GMT+10:00) Vladivostok
84	ASIA_MAGADAN: 'Asia/Magadan'	(GMT+11:00) Magadan, Solomon Is., New Caledonia
85	PACIFIC_KWAJALEIN: 'Pacific/Kwajalein'	(GMT+12:00) Fiji, Marshall Is.
86	PACIFIC_AUCKLAND: 'Pacific/Auckland'	(GMT+12:00) Auckland, Wellington
87	PACIFIC_TONGATAPU: 'Pacific/Tongatapu'	(GMT+13:00) Nuku'alofa

## Syntax



**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](#).

```
// 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

The N/format/i18n module has methods that allows for formatting of strings in international context and for formatting of numbers to currency or number strings.

- [N/format/i18n Module Members](#)
- [N/format/i18n Script Samples](#)

### N/format/i18n Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	format.CurrencyFormatter		Client and server-side scripts	Represents the object that formats the number to currency string.
	format.NumberFormatter		Client and server-side scripts	Represents the object that formats the number to string.
Method	format.spellOut(options)	string	Client and server-side scripts	Creates a string containing the spelled-out version of the specified number in a specified locale.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	format.getCurrencyFormatter(options)	object	Client and server-side scripts	Create the <code>format.CurrencyFormatter</code> object to format numbers to currency strings.
	format.getNumberFormatter(options)	object	Client and server-side scripts	Create the <code>format.NumberFormatter</code> object to format numbers to strings.
Enum	format.NegativeNumberFormat	enum	Client and server-side scripts	Holds the values for the negative number format.
	format.Currency	enum	Client and server-side scripts	Holds the values for the currency code.

## Currency Formatter Object Members

The following members are called on the `format.CurrencyFormatter` object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	CurrencyFormatter.currency	string	Client and server-side scripts	Indicates the currency code.
	CurrencyFormatter.symbol	string	Client and server-side scripts	Indicates the currency symbol.
	CurrencyFormatter.numberFormatter	object	Client and server-side scripts	Contains the <code>format.NumberFormatter</code> object derived from <code>format.CurrencyFormatter</code> with the same number formatting parameters without currency symbol.
	CurrencyFormatter.format(options)	string	Client and server-side scripts	Formats the number to the currency string.

## Number Formatter Object Members

The following members are called on the `format.NumberFormatter` object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	NumberFormatter.groupSeparator	string	Client and server-side scripts	Indicates the group separator.
	NumberFormatter.decimalSeparator	string	Client and server-side scripts	Indicates the decimal separator.
	NumberFormatter.precision	number	Client and server-side scripts	Indicates the precision.
	NumberFormatter.negativeNumberFormat	enum	Client and server-side scripts	Indicates the negative number format.
	NumberFormatter.format(options)	string	Client and server-side scripts	Formats the number to string.

## 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](#).

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample spells out the number 12345 as a string in German, "zwölftausenddreihundertfünfundvierzig".

```
/**  
 * @NApiVersion 2.x  
 */  
  
require(['N/format/i18n'],  
    function(format) {  
        var spellOut = format.spellOut({  
            number: 12345,  
            locale: "DE"  
        });  
  
        log.debug(spellOut);  
    });
```

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample formats a number to string.

```
/**  
 * @NApiVersion 2.x  
 */  
  
require(['N/format/i18n'],  
    function(format) {  
        log.debug("Test of default number formatter:");  
        var numberFormatter = format.getNumberFormatter();  
  
        var gs = numberFormatter.groupSeparator;  
        log.debug("Group separator: " + gs);  
  
        var ds = numberFormatter.decimalSeparator;  
        log.debug("Decimal separator: " + ds);  
  
        var precision = numberFormatter.precision;  
        log.debug("Precision: " + precision);  
  
        var nnf = numberFormatter.negativeNumberFormat;
```

```

    log.debug("Negative Number Format: " + nnf);

    log.debug(numberFormatter.format({number: 12.53}));
    log.debug(numberFormatter.format({number: 12845.22}));
    log.debug(numberFormatter.format({number: -5421}));
    log.debug(numberFormatter.format({number: 0.00}));
    log.debug(numberFormatter.format({number: 0.3456789}));
});

});

```

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample formats numbers to currency strings.

```

/**
 * @NApiVersion 2.x
 */

require(['N/format/i18n'],
function(format) {
    log.debug("Test of currency formatter - EUR:");
    var curFormatter = format.getCurrencyFormatter({currency: "EUR"});

    var curCur = curFormatter.currency;
    log.debug("Currency: " + curCur);

    var numberFormat = curFormatter.numberFormatter;

    var cur3 = curFormatter.symbol;
    log.debug("Currency symbol: " + cur3);

    var c4 = numberFormat.groupSeparator;
    log.debug("Group separator: " + c4);

    var c5 = numberFormat.decimalSeparator;
    log.debug("Decimal separator: " + c5);

    var c6 = numberFormat.precision;
    log.debug("Precision: " + c6);

    var c7 = numberFormat.negativeNumberFormat;
    log.debug("Negative Number Format: " + c7);

    log.debug(curFormatter.format({number: 12.53}));
    log.debug(curFormatter.format({number: -5421}));
    log.debug(curFormatter.format({number: 0.00}));
    log.debug(curFormatter.format({number: 0.3456789}));
});

});

```

## format.CurrencyFormatter



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	The object that formats the number to currency string.
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/format/i18n Module</a>
<b>Methods and Properties</b>	<a href="#">N/format/i18n Module Members</a>
<b>Since</b>	2019.2

## 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/format/i18n Script Samples](#).

```
// Add additional code
...
require(['N/format/i18n'],
    function(format) {
        var curFormatter = format.getCurrencyFormatter({currency: "USD"});
});...
// Add additional code
```

## CurrencyFormatter.currency

<b>Property Description</b>	Describes the currency code.
<b>Type</b>	string (read-only)
<b>Module</b>	<a href="#">N/format/i18n Module</a>
<b>Parent Object</b>	<a href="#">format.CurrencyFormatter</a>
<b>Sibling Object Members</b>	<a href="#">N/format/i18n Module Members</a>
<b>Since</b>	2019.2

## 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/format/i18n Script Samples](#).

```
// Add additional code
```

```

...
require(['N/format/i18n'],
  function(format) {
    var curFormatter = format.getCurrencyFormatter({currency: "USD"});
    var curCur = curFormatter.currency;
    log.debug(curCur);
});...
// Add additional code

```

## CurrencyFormatter.symbol

<b>Property Description</b>	Describes the symbol of the currency code.
<b>Type</b>	string (read-only)
<b>Module</b>	N/format/i18n Module
<b>Parent Object</b>	format.CurrencyFormatter
<b>Sibling Object Members</b>	<a href="#">N/format/i18n Module Members</a>
<b>Since</b>	2019.2

## 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/format/i18n Script Samples](#).

```

// Add additional code
...
require(['N/format/i18n'],
  function(format) {
    var curFormatter = format.getCurrencyFormatter({currency: "USD"});
    var curSymbol = curFormatter.symbol;
    log.debug(curSymbol);
});...
// Add additional code

```

## CurrencyFormatter.numberFormatter

<b>Property Description</b>	Contains the <a href="#">format.NumberFormatter</a> object derived from <a href="#">format.CurrencyFormatter</a> with the same number formatting parameters without currency symbol.
<b>Type</b>	string (read-only)
<b>Module</b>	N/format/i18n Module
<b>Parent Object</b>	format.CurrencyFormatter
<b>Sibling Object Members</b>	<a href="#">N/format/i18n Module Members</a>
<b>Since</b>	2019.2

## 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/format/i18n Script Samples](#).

```
// Add additional code
...
require(['N/format/i18n'],
    function(format) {
        var curFormatter = format.getCurrencyFormatter({currency: "USD"});
        var numberFormatter = curFormatter.numberFormatter;

        // now numberFormatter object can be used
        log.debug(numberFormatter.format({number: -12.5366}));
});...
// Add additional code
```

## CurrencyFormatter.format(options)

<b>Method Description</b>	Formats the number to the currency string.
<b>Returns</b>	String
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10
<b>Module</b>	<a href="#">N/format/i18n Module</a>
<b>Methods and Properties</b>	<a href="#">N/format/i18n Module Members</a>
<b>Since</b>	2019.2

## Parameters



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

Parameter	Type	Required / Optional	Description
options.number	number	required	The number to be formatted

## Syntax



**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/i18n Script Samples](#).

```
// Add additional code
...
require(['N/format/i18n'],
    function(format) {
        var curFormatter = format.getCurrencyFormatter({currency: "USD"});
```

```

        log.debug(curFormatter.format({number: 12.53}));
    });
}

// Add additional code

```

## format.NumberFormatter

<b>Object Description</b>	Object that formats number to string.
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/format/i18n Module</a>
<b>Methods and Properties</b>	<a href="#">N/format/i18n Module Members</a>
<b>Since</b>	2019.2

## 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/format/i18n Script Samples](#).

```

// Add additional code
...
require(['N/format/i18n'],
    function(format) {
        var numFormatter1 = format.getNumberFormatter(); // no parameter given -> default number formatter object
        returned
        var numFormatter2 = format.getNumberFormatter({
            groupSeparator: " ",
            decimalSeparator: ",",
            precision: 2,
            negativeNumberFormat: format.NegativeNumberFormat_MINUS});
        // all parameters defined

        // here number formatters can be used
        log.debug(numFormatter1.format({number: 12.53}));
        log.debug(numFormatter2.format({number: 12845.22}));
    });
}

// Add additional code

```

## NumberFormatter.groupSeparator

<b>Property Description</b>	Indicates the group separator.
<b>Type</b>	string (read-only)
<b>Module</b>	<a href="#">N/format/i18n Module</a>
<b>Parent Object</b>	<a href="#">format.CurrencyFormatter</a>
<b>Sibling Object Members</b>	<a href="#">N/format/i18n Module Members</a>

Since	2019.2
-------	--------

## 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/format/i18n Script Samples](#).

```
// Add additional code
...
require(['N/format/i18n'],
    function(format) {
        var numFormatter = format.getNumberFormatter({
            groupSeparator: " ",
            decimalSeparator: ",",
            precision: 2,
            negativeNumberFormat: format.NegativeNumberFormat_MINUS});
        var groupSep = numFormatter.groupSeparator;
    });
// Add additional code
```

## NumberFormatter.decimalSeparator

<b>Property Description</b>	Indicates the decimal separator.
<b>Type</b>	string (read-only)
<b>Module</b>	N/format/i18n Module
<b>Parent Object</b>	format.CurrencyFormatter
<b>Sibling Object Members</b>	<a href="#">N/format/i18n Module Members</a>
<b>Since</b>	2019.2

## 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/format/i18n Script Samples](#).

```
// Add additional code
...
require(['N/format/i18n'],
    function(format) {
        ...
        var numFormatter = format.getNumberFormatter({
            groupSeparator: " ",
            decimalSeparator: ",",
            precision: 2,
            negativeNumberFormat: format.NegativeNumberFormat_MINUS});
        var decimalSep = numFormatter.decimalSeparator;
    });
// Add additional code
```

## NumberFormatter.precision

<b>Property Description</b>	Indicates the precision.
<b>Type</b>	number (read-only)
<b>Module</b>	N/format/i18n Module
<b>Parent Object</b>	format.CurrencyFormatter
<b>Sibling Object Members</b>	<a href="#">N/format/i18n Module Members</a>
<b>Since</b>	2019.2

### 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/format/i18n Script Samples](#).

```
// Add additional code
...
require(['N/format/i18n'],
    function(format) {
        var numFormatter = format.getNumberFormatter({
            groupSeparator: " ",
            decimalSeparator: ",",
            precision: 2,
            negativeNumberFormat: format.NegativeNumberFormat_MINUS});
        var precision = numFormatter.precision;
});...
// Add additional code
```

## NumberFormatter.negativeNumberFormat

<b>Property Description</b>	Indicates the negative number format.
<b>Type</b>	enum
<b>Module</b>	N/format/i18n Module
<b>Parent Object</b>	format.CurrencyFormatter
<b>Sibling Object Members</b>	<a href="#">N/format/i18n Module Members</a>
<b>Since</b>	2019.2

### 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/format/i18n Script Samples](#).

```
// Add additional code
...
require(['N/format/i18n'],
```

```

function(format) {
    var numFormatter = format.getNumberFormatter({
        groupSeparator: " ",
        decimalSeparator: ",",
        precision: 2,
        negativeNumberFormat: format.NegativeNumberFormat_MINUS});
    var negNumFormat = numFormatter.negativeNumberFormat;
}
};

// Add additional code

```

## NumberFormatter.format(options)

<b>Method Description</b>	Format number to the number string.
<b>Returns</b>	String
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10
<b>Module</b>	<a href="#">N/format/i18n Module</a>
<b>Methods and Properties</b>	<a href="#">N/format/i18n Module Members</a>
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.number	number	required	The number to be formatted

## Syntax

 **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/i18n Script Samples](#).

```

// Add additional code
...
require(['N/format/i18n'],
function(format) {
    var numFormatter = format.getNumberFormatter({
        groupSeparator: " ",
        decimalSeparator: ",",
        precision: 2,
        negativeNumberFormat: format.NegativeNumberFormat_MINUS});
    // all parameters defined
    log.debug(numFormatter.format({number: 12845.22}));
}
};

// Add additional code

```

## format.spellOut(options)

<b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.
<b>Method Description</b> 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> .
<b>Returns</b> String
<b>Supported Script Types</b> Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b> None
<b>Module</b> <a href="#">N/format/i18n Module</a>
<b>Methods and Properties</b> <a href="#">N/format/i18n Module Members</a>
<b>Since</b> 2019.1

### Parameters

<b>Note:</b> The options parameter is a JavaScript object.
<b>Parameter</b>
<b>Type</b>
<b>Required / Optional</b>
<b>Description</b>
<b>Since</b>
<code>options.number</code>
number
required
The number to be spelled out in a string.
2019.1
<code>options.locale</code>
string
required
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> .
2019.1

## format.getCurrencyFormatter(options)

<b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.
<b>Method Description</b> Create <code>format.CurrencyFormatter</code> object to format numbers into currency strings.
<b>Returns</b> Object
<b>Supported Script Types</b> Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b> 10
<b>Module</b> <a href="#">N/format/i18n Module</a>

<b>Methods and Properties</b>	N/format/i18n Module Members
<b>Since</b>	2019.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.currency	string	required	Code of the currency that is used by formatter.	2019.2

## Errors

Error Code	Thrown If
MISSING_REQD_ARGUMENT	Currency parameter is missing
SSS_INVALID_CURRENCY	The currency is not valid
SSS_INVALID_TYPE_ARG	The parameter type is wrong

## format.getNumberFormatter(options)

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Create <a href="#">format.NumberFormatter</a> object to format numbers into strings.
<b>Returns</b>	Object
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10
<b>Module</b>	<a href="#">N/format/i18n Module</a>
<b>Methods and Properties</b>	<a href="#">N/format/i18n Module Members</a>
<b>Since</b>	2019.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.groupSeparator	string	optional	Indicates the group separator.	2019.2
options.decimalSeparator	string	optional	Indicates the decimal separator.	2019.2
options.precision	number	optional	Indicates the precision.	2019.2

Parameter	Type	Required / Optional	Description	Since
options.negativeNumberFormat	enum	optional	Indicates the negative number format.	2019.2

## format.NegativeNumberFormat

<b>Enum Description</b>	Holds the values for the negative number format.
	<p><b>Note:</b> 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>
<b>Type</b>	enum
<b>Module</b>	<a href="#">N/format/i18n Module</a>
<b>Sibling Module Members</b>	<a href="#">N/format/i18n Module Members</a>
<b>Since</b>	2019.2

## Values

Value
BRACKETS
MINUS

## Syntax

**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/i18n Script Samples](#).

```
// Add additional code
...
require(['N/format/i18n'],
    function(format) {
        var numFormatterM = format.getNumberFormatter({
            groupSeparator: " ", decimalSeparator: ",",
            precision: 2,
            negativeNumberFormat: format.NegativeNumberFormat.MINUS});

        var numFormatterB = format.getNumberFormatter({
            groupSeparator: " ",
            decimalSeparator: ",",
            precision: 2,
            negativeNumberFormat: format.NegativeNumberFormat.BRACKETS});

});...
// Add additional code
```

## format.Currency

<b>Enum Description</b>	Holds the values for the currency code.
	<p><b>Note:</b> 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>
<b>Type</b>	enum
<b>Module</b>	N/format/i18n Module
<b>Sibling Module Members</b>	N/format/i18n Module Members
<b>Since</b>	2019.1

## Value

The currency values depend on the company. Examples of currency value include:

- USD
- CAD
- EUR
- GBP

## Syntax

**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/i18n Script Samples](#).

```
// Add additional code
...
require(['N/format/i18n'],
    function(format) {
        log.debug("List of valid currencies:");
        for (var currency in format.Currency) {
            log.debug(currency);
        }
});...
// Add additional code
```

## N/http Module

**Note:** The content in this help topic pertains to SuiteScript 2.0.

Use the N/http module to make HTTP calls from server or client scripts. For client scripts, this module also provides the ability to make cross-domain HTTP requests using NetSuite servers as proxies.

All HTTP content types are supported.



**Note:** The N/http module does not accept the HTTPS protocol. Use the [N/https Module](#) for that purpose.

- [HTTP Header Information](#)
- [N/http Module Members](#)
- [ClientResponse Object Members](#)
- [ServerRequest Object Members](#)
- [ServerResponse Object Members](#)
- [N/http Module Script Samples](#)



**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 Header Information

HTTP headers can be used to pass additional information with an HTTP request or response. Each HTTP header consists of its case-insensitive name followed by a colon (:), then by its value (without line breaks). For a general list of all HTTP headers, visit <http://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>.

HTTP headers are accessible in `http.ClientResponse`, `http.ServerRequest`, and `http.ServerResponse`.

In NetSuite, some headers are not supported. These are listed below as either general HTTP headers or Suitelet response headers.

## General HTTP Header Blacklist

Be aware that certain headers cannot be set manually when using the N/http module methods. If a script attempts to set values for any of the following headers, the values are discarded. These headers are listed in the following table.

<ul style="list-style-type: none"> <li>▪ Connection</li> <li>▪ Content-Length</li> <li>▪ Host</li> <li>▪ Trailer</li> </ul>	<ul style="list-style-type: none"> <li>▪ Transfer-Encoding</li> <li>▪ Upgrade</li> <li>▪ Via</li> </ul>
---	---

## 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 listed in the following table.

<ul style="list-style-type: none"> <li>▪ Access-Control-Allow-Origin</li> <li>▪ Allow</li> <li>▪ Connection</li> <li>▪ Content-Length</li> <li>▪ Content-Location</li> <li>▪ Content-MD5</li> </ul>	<ul style="list-style-type: none"> <li>▪ Date</li> <li>▪ Location</li> <li>▪ Proxy-Authenticate</li> <li>▪ Retry-After</li> <li>▪ Server</li> <li>▪ Trailer</li> </ul>	<ul style="list-style-type: none"> <li>▪ Warning</li> <li>▪ WWW-Authenticate</li> </ul>
---	--	---

Content-Type	Via	
--------------	-----	--

## N/http Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">http.ClientResponse</a>	Object (read-only)	Server scripts	Encapsulates the response to an HTTP client request (e.g., <code>http.get(options)</code> ).
	<a href="#">http.ServerRequest</a>	Object (read-only)	Server scripts	Encapsulates the HTTP request information sent to an HTTP server. For example, a request received by a Suitelet or RESTlet.
	<a href="#">http.ServerResponse</a>	Object	Server scripts	Encapsulates the response from an HTTP server to an HTTP request. For example, a response from a Suitelet or RESTlet.
Method	<a href="#">http.delete(options)</a>	<a href="#">http.ClientResponse</a>	Server scripts	Sends an HTTP DELETE request and returns the response.
	<a href="#">http.delete.promise(options)</a>	<a href="#">http.ClientResponse</a>	Client scripts	Sends an HTTP DELETE request asynchronously and returns the response.
	<a href="#">http.get(options)</a>	<a href="#">http.ClientResponse</a>	Server scripts	Sends an HTTP GET request and returns the response.
	<a href="#">http.get.promise(options)</a>	<a href="#">http.ClientResponse</a>	Client scripts	Sends an HTTP GET request asynchronously and returns the response.
	<a href="#">http.post(options)</a>	<a href="#">http.ClientResponse</a>	Server scripts	Sends an HTTP POST request and returns the response.
	<a href="#">http.post.promise(options)</a>	<a href="#">http.ClientResponse</a>	Client scripts	Sends an HTTP POST request asynchronously and returns the response.
	<a href="#">http.put(options)</a>	<a href="#">http.ClientResponse</a>	Server scripts	Sends an HTTP PUT request and returns the response.
	<a href="#">http.put.promise(options)</a>	<a href="#">http.ClientResponse</a>	Client scripts	Sends an HTTP PUT request asynchronously and returns the response.
	<a href="#">http.request(options)</a>	<a href="#">http.ClientResponse</a>	Server scripts	Sends an HTTP request and returns the response.
	<a href="#">http.request.promise(options)</a>	<a href="#">http.ClientResponse</a>	Client scripts	Sends an HTTP request asynchronously and returns the response.
Enum	<a href="#">http.CacheDuration</a>	enum	Server scripts	Holds the string values for supported cache durations. This enum is used to set the value of the <code>ServerResponse.setCdnCacheable(options)</code> property.
	<a href="#">http.Method</a>	enum	Serverscripts	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> .
	<a href="#">http.RedirectType</a>	enum	Server scripts	Holds the string values for supported NetSuite resources that you can redirect to. This enum is used

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
				to set the value of the <code>type</code> argument for <a href="#">ServerResponse.sendRedirect(options)</a> .

## ClientResponse Object Members

The following members are called on the [http.ClientResponse](#) Object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	<a href="#">ClientResponse.body</a>	string (read-only)	Server scripts	The client response body.
	<a href="#">ClientResponse.code</a>	number (read-only)	Server scripts	The client response code.
	<a href="#">ClientResponse.headers</a>	Object (read-only)	Server scripts	The client response headers.

## ServerRequest Object Members

The following members are called on the [http.ServerRequest](#) Object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">ServerRequest.getLineCount(options)</a>	number	Server scripts	Returns the number of lines in a sublist.
	<a href="#">ServerRequest.getSublistValue(options)</a>	string	Server scripts	Returns the value of a sublist line item.
Property	<a href="#">ServerRequest.body</a>	string (read-only)	Server scripts	The server request body.
	<a href="#">ServerRequest.files</a>	Object (read-only)	Server scripts	The server request files.
	<a href="#">ServerRequest.headers</a>	Object (read-only)	Server scripts	The server request headers.
	<a href="#">ServerRequest.method</a>	<a href="#">http.Method</a> enum	Server scripts	The server request HTTP method.
	<a href="#">ServerRequest.parameters</a>	Object (read-only)	Server scripts	The server request parameters.
	<a href="#">ServerRequest.url</a>	string (read-only)	Server scripts	The server request URL.

## ServerResponse Object Members

The following members are called on the [http.ServerResponse](#) Object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">ServerResponse.addHeader(options)</a>	void	Server scripts	Adds a header to the response.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	ServerResponse.getHeader(options)	string   string[]	Server scripts	Returns the value of a response header.
	ServerResponse.renderPdf(options)	void	Server scripts	Generates and renders a PDF directly to the response.
	ServerResponse.sendRedirect(options)	void	Server scripts	Sets the redirect URL by resolving to a NetSuite resource.
	ServerResponse.setCdnCacheable(options)	void	Server scripts	Sets CDN caching for a period of time.
	ServerResponse.setHeader(options)	void	Server scripts	Sets the value of a response header.
	ServerResponse.write(options)	void	Server scripts	Writes information (text, xml, html) to the response.
	ServerResponse.writeFile(options)	void	Server scripts	Writes a file to the response.
	ServerResponse.writeLine(options)	void	Server scripts	Writes line information (text, xml, html) to the response.
	ServerResponse.writePage(options)	void	Server scripts	Generates a page.
Property	ServerResponse.headers	Object (read-only)	Server scripts	The server response headers.

## N/http Module Script Samples

### Example 1

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to use an HTTP GET request for a URL.

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

require(['N/http'], function(http) {
    function sendGetRequest() {
        var response = http.get({
            url: 'http://www.google.com'
        });
    }
})
```

```
    sendGetRequest();
});
```

## Example 2

**Note:** This script sample uses the **define** function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the **require** function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

**Important:** The value used in this sample for the entity field is a placeholder. Before using this sample, replace the entity field value with a valid value from your NetSuite account. If you run a script with an invalid value, an error may occur.

The following sample shows how to redirect to a new sales order record and set entity to 6. (Assuming there is an entity with number 6, if there's not, then the entity will remain blank.)

```
/**  
 * @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

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the response to an HTTP client request (i.e., the return type for <a href="#">http.delete(options)</a> , <a href="#">http.get(options)</a> , <a href="#">http.post(options)</a> , <a href="#">http.put(options)</a> , <a href="#">http.request(options)</a> , and corresponding promise methods).  This object is read-only.  For a complete list of this object's properties, see <a href="#">ClientResponse Object Members</a> .
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/http Module</a>
<b>Since</b>	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 Samples](#).

```
// Add additional code
...
var clientResponse = http.get({
    url: 'http://www.google.com'
});
...
// Add additional code
```

## ClientResponse.body



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The client response body. This property is read-only.
<b>Type</b>	string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/http Module</a>
<b>Parent Object</b>	<a href="#">http.ClientResponse</a>
<b>Sibling Object Members</b>	<a href="#">ClientResponse Object Members</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax



**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 Samples](#).

```
// Add additional code
...
var response = http.get({
    url: 'http://www.google.com'
});
log.debug({
    title: 'Client Response Body',
    details: response.body
});
```

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

## ClientResponse.code



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The client HTTP response or status code.  This property is read-only.
<b>Type</b>	number
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/http Module</a>
<b>Parent Object</b>	<a href="#">http.ClientResponse</a>
<b>Sibling Object Members</b>	<a href="#">ClientResponse Object Members</a>
<b>Since</b>	2015.2

### Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

### Syntax



**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 Samples](#).

```
// Add additional code
...
var response = http.get({
    url: 'http://www.google.com'
});
log.debug({
    title: 'Client Response Code',
    details: response.code
});
...
// Add additional code
```

## ClientResponse.headers



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The response header or headers.
-----------------------------	---------------------------------

	This property is read-only. For more information, see <a href="#">HTTP Header Information</a> .
<b>Type</b>	Object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/http Module
<b>Parent Object</b>	http.ClientResponse
<b>Sibling Object Members</b>	<a href="#">ClientResponse Object Members</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax

 **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 Samples](#).

```
// Add additional code
...
var response = http.get({
    url: 'http://www.google.com'
});
log.debug({
    title: 'Client Response Header',
    details: response.headers
});
...
// Add additional code
```

## http.ServerRequest

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the HTTP request information set 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 <a href="#">ServerRequest Object Members</a> .
<b>Supported Script Types</b>	Server scripts

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/http Module
<b>Since</b>	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 Samples](#).

```
// Add additional code
...
serverRequest.getLineCount({
    group: 'sublistId'
});
...
// Add additional code
```

## ServerRequest.getLineCount(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the number of lines in a sublist.
<b>Returns</b>	The number of lines in a sublist as a number.
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/http Module
<b>Parent Object</b>	http.ServerRequest
<b>Sibling Object Members</b>	ServerRequest Object Members
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.group	string	required	The sublist internal ID.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required 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/http Module Script Samples](#).

```
// Add additional code
...
serverRequest.getLineCount({
  group: 'sublistId'
});
...
// Add additional code
```

## ServerRequest.getSublistValue(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the value of a sublist line item.
<b>Returns</b>	The value of the sublist line item as a string.
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/http Module</a>
<b>Parent Object</b>	<a href="#">http.ServerRequest</a>
<b>Sibling Object Members</b>	<a href="#">ServerRequest Object Members</a>
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.group	string	required	The sublist internal ID.	2015.2
options.line	string	required	The sublist line number.  <b>Note:</b> Sublist index starts at 0.	2015.2
options.name	string	required	The sublist line item ID (name of the field).	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required 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/http Module Script Samples](#).

```
// Add additional code
...
serverRequest.getSublistValue({
  group: 'item',
  name: 'amount',
  line: '2'
});
...
// Add additional code
```

## ServerRequest.body

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The server request body. This property is read-only.
<b>Type</b>	string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/http Module</a>
<b>Parent Object</b>	<a href="#">http.ServerRequest</a>
<b>Sibling Object Members</b>	<a href="#">ServerRequest Object Members</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax

**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 Samples](#).

```
// Add additional code
...
log.debug({
  title: 'Server Request Body',
  details: request.body
});
...
```

```
// Add additional code
```

## ServerRequest.files

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The server request files. This property is read-only.
<b>Type</b>	Object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/http Module
<b>Parent Object</b>	<a href="#">http.ServerRequest</a>
<b>Sibling Object Members</b>	<a href="#">ServerRequest Object Members</a>
<b>Since</b>	2015.2

### Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

### 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 Samples](#).

```
// Add additional code
...
log.debug({
    title: 'Server Request Files',
    details: request.files
});
...
// Add additional code
```

```
var file = request.files['file_id'];
```

## ServerRequest.headers

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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
-----------------------------	---

	<p>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).</p> <p>This property is read-only.</p>
<p><b>Important:</b> 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.</p>	
	<p>For more information, see <a href="#">HTTP Header Information</a>.</p>
<b>Type</b>	Object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/http Module</a>
<b>Parent Object</b>	<a href="#">http.ServerRequest</a>
<b>Sibling Object Members</b>	<a href="#">ServerRequest Object Members</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax

**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 Samples](#).

```
// Add additional code
...
log.debug({
  title: 'Server Request Headers',
  details: request.headers
});
...
// Add additional code
```

## ServerRequest.method

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The server request HTTP method.
-----------------------------	---------------------------------

	This property is read-only.
<b>Type</b>	<a href="#">http.Method</a>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/http Module</a>
<b>Parent Object</b>	<a href="#">http.ServerRequest</a>
<b>Sibling Object Members</b>	<a href="#">ServerRequest Object Members</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax

**⚠ 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 Samples](#).

```
// Add additional code
...
log.debug({
    title: 'Server Request Method',
    details: request.method
});
...
// Add additional code
```

## ServerRequest.parameters

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The server request parameters.  This property is read-only.
<b>Type</b>	Object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/http Module</a>
<b>Parent Object</b>	<a href="#">http.ServerRequest</a>
<b>Sibling Object Members</b>	<a href="#">ServerRequest Object Members</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax

**⚠ 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 Samples](#).

```
// Add additional code
...
log.debug({
    title: 'Server Request Parameters',
    details: request.parameters
});
...
// Add additional code
```

## ServerRequest.url

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The server request URL. This property is read-only.
<b>Type</b>	string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/http Module</a>
<b>Parent Object</b>	<a href="#">http.ServerRequest</a>
<b>Sibling Object Members</b>	<a href="#">ServerRequest Object Members</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax

**⚠ 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 Samples](#).

```
log.debug({
    title: 'Server Request URL',
```

```

    details: request.url
});
...
// Add additional code

```

## http.ServerResponse



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the response from an HTTP server to an HTTP request. For example, a response from a Suitelet or RESTlet.  For a complete list of this object's methods and properties, see <a href="#">ServerResponse Object Members</a> .
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/http Module</a>
<b>Since</b>	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 Samples](#).

```

// Add additional code
...
serverResponse.addHeader({
    name: 'Accept-Language',
    value: 'en-us',
});
...
// Add additional code

```

## ServerResponse.addHeader(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a header to the response.  If the same header has already been set, this method adds another line for that header. For example:
	<pre>{Vary: ['Accept-Language', 'Accept-Encoding']}</pre>
	For more information, see <a href="#">HTTP Header Information</a> .
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Governance</b>	None
<b>Module</b>	N/http Module
<b>Parent Object</b>	http.ServerResponse
<b>Sibling Object Members</b>	ServerResponse Object Members
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	The name of the header.	2015.2
options.value	string	required	The value used to set the header.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing.
SSS_INVALID_HEADER	One or more headers are not valid.	The header name or value is invalid.

## Syntax

 **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 Samples](#).

```
// Add additional code
...
serverResponse.addHeader({
    name: 'Accept-Language',
    value: 'en-us',
});
...
// Add additional code
```

## ServerResponse.getHeader(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the value or values of a response header. If multiple values are assigned to the header name, the values are returned as an Array.  For more information, see <a href="#">HTTP Header Information</a> .
<b>Returns</b>	string   string[]
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Governance</b>	None
<b>Module</b>	N/http Module
<b>Parent Object</b>	http.ServerResponse
<b>Sibling Object Members</b>	ServerResponse Object Members
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	The name of the header.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required 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/http Module Script Samples](#).

```
// Add additional code
...
serverResponse.getHeader({
    name: 'Accept-Language'
});
...
// Add additional code
```

## ServerResponse.sendRedirect(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the redirect URL by resolving to a NetSuite resource.  For example, you could use this method to redirect to a new sales order page for a particular entity.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/http Module
<b>Parent Object</b>	http.ServerResponse

<b>Sibling Object Members</b>	ServerResponse Object Members
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.identifier	number   string	required	<p>The primary ID for this resource. The value you use varies depending on the value of options.type, as follows:</p> <ul style="list-style-type: none"> <li>■ MEDIA_ITEM — Use the internal ID of a file stored in the NetSuite File Cabinet.</li> <li>■ RECORD — Use the <a href="#">record.Type</a> enum to identify the appropriate record type.</li> <li>■ RESTLET — Use the script ID from the script record of the appropriate RESTlet.</li> <li>■ SUITELET — Use the script ID from the script record of the appropriate Suitelet.</li> <li>■ TASK_LINK — Use the appropriate Task ID. Supported IDs are listed in <a href="#">Task IDs</a>.</li> </ul>	2015.2
options.type	<a href="#">http.RedirectType</a>	required	The type of resource redirected to.	2015.2
options.editMode	boolean <b>true</b>   <b>false</b>	optional	<p>Applicable when redirecting to a record resource.</p> <p>Specifies whether to return a URL for a record in edit mode or view mode.</p> <p>If set to <b>true</b>, returns the record in edit mode. If set to <b>false</b>, returns the record in view mode.</p> <p>The default value is <b>false</b>.</p>	2015.2
options.id	number   string	optional	The secondary ID for this resource. If the options.type parameter is set to SUITELET or RESTLET, use the deployment ID. If the options.type parameter is set to RECORD, you can use the internal ID of a specific record instance.	2015.2
options.parameters	Object	optional	Additional URL parameters as name/value pairs.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing. Note that this error is thrown if an enum is misspelled within

Error Code	Message	Thrown If
		a script. For example, you see this error if you use http.RedirectType.TASKLINK instead of http.RedirectType.TASK_LINK in the options.type field.
SSS_INVALID_URL_CATEGORY	The options.type: {type} is not valid. Please use <a href="#">http.RedirectType</a> for supported types.	The script uses an unrecognizable string value for the options.type parameter. To avoid this error, use <a href="#">http.RedirectType</a> .
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 task link, 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 Samples](#).

```
// Add additional code
...
myServerResponseObj.sendRedirect({
    type: http.RedirectType.RECORD,
    identifier: record.Type.SALES_ORDER,
    parameters: {entity: 8}
});
...
// Add additional code
```

## ServerResponse.setHeader(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value of a response header. For more information, see <a href="#">HTTP Header Information</a> .
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None

<b>Module</b>	N/http Module
<b>Parent Object</b>	http.ServerResponse
<b>Sibling Object Members</b>	ServerResponse Object Members
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	The name of the header.	2015.2
options.value	string	required	The value used to set the header.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing.
SSS_INVALID_HEADER	One or more headers are not valid.	The header name or value is invalid.

## Syntax

 **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 Samples](#).

```
// Add additional code
...
serverResponse.setHeader({
    name: 'Accept-Language',
    value: 'en-us',
});
...
// Add additional code
```

## ServerResponse.renderPdf(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Generates and renders a PDF directly to the response.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	N/http Module

<b>Parent Object</b>	http.ServerResponse
<b>Sibling Object Members</b>	ServerResponse Object Members
<b>Since</b>	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 missing.

## 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 Samples](#).

```
/*
 *@NApiVersion 2.0
 *@NScriptType suitelet
 */
define(['N/xml'], function(xml){
    return {
        onRequest: function(context){
            var xml = "<?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\" " + "src=\"NetSuiteFonts/verdana.ttf\"
\" " + "src-bold=\"NetSuiteFonts/verdanab.ttf\" " + "src-italic=\"NetSuiteFonts/verdanai.ttf\" " + "src-bolditalic=\"NetSuiteFonts/verdanabi.ttf\" " + "bytes=\"2\"/>\n" +
                "</head>\n" +
                "<body font-family=\"russianfont\" font-size=\"18\">\nРусский текст</body>\n" +
                "</pdf>";
            context.response.renderPdf(xml);
        }
    });
});
```

## ServerResponse.setCdnCacheable(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets CDN caching for a period of time.
---------------------------	--

<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/http Module</a>
<b>Parent Object</b>	<a href="#">http.ServerResponse</a>
<b>Sibling Object Members</b>	<a href="#">ServerResponse Object Members</a>
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.type	enum	required	The value of the caching duration. Set using the <a href="#">http.CacheDuration</a> enum.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required 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/http Module Script Samples](#).

```
// Add additional code
...
serverResponse.setCdnCacheable({
    type: http.CacheDuration.MAX
});
...
// Add additional code
```

## ServerResponse.write(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Writes information (text, xml, html) to the response.
	<b>Note:</b> This method accepts only strings. To pass in a file, you can use <a href="#">ServerResponse.writeFile(options)</a> .
<b>Returns</b>	void

<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/http Module
<b>Parent Object</b>	<a href="#">http.ServerResponse</a>
<b>Sibling Object Members</b>	<a href="#">ServerResponse Object Members</a>
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.output	string	required	The string being written.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing.
WRONG_PARAMETER_TYPE	{param name}	The value input for options.output is not a string.

## Syntax

**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 Samples](#).

```
// Add additional code
...
serverResponse.write({
    output: 'Hello World'
});
...
// Add additional code
```

## ServerResponse.writeFile(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Writes a file to the response.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/http Module</a>
<b>Parent Object</b>	<a href="#">http.ServerResponse</a>
<b>Sibling Object Members</b>	<a href="#">ServerResponse Object Members</a>
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.file	<a href="#">file.File</a>	required	A <a href="#">file.File</a> Object that encapsulates the file to be written.	2015.2
options.isInline	boolean <b>true</b>   <b>false</b>	optional	If <b>true</b> , the file is inline. The default value is <b>false</b> .	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing.
WRONG_PARAMETER_TYPE	{param name}	The value input for options.file is not a <a href="#">file.File</a> 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/http Module Script Samples](#).

```
// Add additional code
...
serverResponse.writeFile({
  file: myFileObj,
  isInline: true
});
...
// Add additional code
```

## ServerResponse.writeLine(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Writes line information (text, xml, html) to the response.
---------------------------	--

<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/http Module</a>
<b>Parent Object</b>	<a href="#">http.ServerResponse</a>
<b>Sibling Object Members</b>	ServerResponse Object Members
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.output	string	required	The string being written.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing.
WRONG_PARAMETER_TYPE	{param name}	The value input for options.output is not a string.

## Syntax

 **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 Samples](#).

```
// Add additional code
...
serverResponse.writeLine({
    output: 'this is a sample string'
});
...
// Add additional code
```

## ServerResponse.writePage(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Generates a page.
<b>Returns</b>	void

<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/http Module
<b>Parent Object</b>	http.ServerResponse
<b>Sibling Object Members</b>	<a href="#">ServerResponse Object Members</a>
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.pageObject	<a href="#">serverWidget.Assistant</a>   <a href="#">serverWidget.Form</a>   <a href="#">serverWidget.List</a>	required	A standalone page Object in the form of an assistant, form, or list.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required 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/http Module Script Samples](#).

```
// Add additional code
...
var myPageObj = serverWidget.createList({
    title: 'Simple List'
});

ServerResponse.writePage({
    pageObject: myPageObj
});
...
// Add additional code
```

## ServerResponse.headers



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The server response headers.
-----------------------------	------------------------------

	<p>This property is read-only.</p> <p>For more information, see <a href="#">HTTP Header Information</a>.</p>
<b>Type</b>	<p>Object</p> <p>If multiple values are assigned to one header name, the values are returned as an array. For example:</p> <pre>{Vary: ['Accept-Language', 'Accept-Encoding']}</pre>
<b>Supported Script Types</b>	<p>Server scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Module</b>	<a href="#">N/http Module</a>
<b>Parent Object</b>	<a href="#">http.ServerResponse</a>
<b>Sibling Object Members</b>	<a href="#">ServerResponse Object Members</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax



**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 Samples](#).

```
// Add additional code
...
log.debug({
    title: "Server Response Headers",
    details: ServerResponse.headers
});
...
// Add additional code
```

## http.get(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTP GET request.
<b>Returns</b>	<a href="#">http.ClientResponse</a>
<b>Supported Script Types</b>	<p>Server scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>

<b>Governance</b>	10 units
<b>Module</b>	N/http Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.url	string	required	The HTTP URL being requested.	2015.2
options.headers	Object	optional	The HTTP headers.  For more information, see <a href="#">HTTP Header Information</a> .	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing.

## Syntax



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

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
var response = http.get({
    url: 'http://www.google.com',
    headers: headerObj
});
...
// Add additional code
```

## http.get.promise(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTP GET request asynchronously.
---------------------------	---

	<b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">http.get(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">http.get(options)</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	10 units
<b>Module</b>	N/http Module
<b>Since</b>	2015.2

## Syntax

**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
http.get.promise({
    url: 'http://www.google.com',
    headers: headerObj
})
    .then(function(response){
        log.debug({
            title: 'Response',
            details: response
        });
    })
    .catch(function onRejected(reason) {
        log.debug({
            title: 'Invalid Get Request: ',
            details: reason
        });
    })
...
// Add additional code
```

## http.delete(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTP DELETE request.
---------------------------	-------------------------------

	 <b>Important:</b> 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.
<b>Returns</b>	<a href="#">http.ClientResponse</a>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/http Module</a>
<b>Since</b>	2015.2

## Parameters

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

 **Note:** This method does not include an `options.body` parameter. Postdata is not required when the HTTP method is a DELETE request.

Parameter	Type	Required / Optional	Description	Since
options.url	string	required	The HTTP URL being requested	2015.2
options.headers	Object	optional	The HTTP headers.  For more information, see <a href="#">HTTP Header Information</a> .	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required 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/http Module Script Samples](#).

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
var response = http.delete({
    url: 'http://www.mytestwebsite.com',
```

```

    headers: headerObj
});
...
// Add additional code

```

## http.delete.promise(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTP DELETE request asynchronously.   <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">http.delete(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">http.delete(options)</a>
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/http Module</a>
<b>Since</b>	2015.2

### Syntax

 **Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```

// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
http.delete.promise({
    url: 'http://www.mytestwebsite.com',
    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.post(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTP POST request.
	 <b>Important:</b> 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.
<b>Returns</b>	<a href="#">http.ClientResponse</a>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/http Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.url	string	required	The HTTP URL being requested	2015.2
options.body	string   Object	required	The POST data.	2015.2
options.headers	Object	optional	The HTTP headers. For more information, see <a href="#">HTTP Header Information</a> .	2015.2

### Errors

Error Code	Message	Thrown If
SSS_INVALID_URL	The URL must be a fully qualified HTTP/HTTPS URL	An incorrect protocol is used, such as using HTTP within the HTTPS module.

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing.
SSS_REQUEST_LOOP_DETECTED	This 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.	A script is calling back into itself recursively via an HTTP/HTTPS request.

## Syntax



**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 Samples](#).

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
var response = http.post({
    url: 'http://www.google.com',
    body: 'My POST Data',
    headers: headerObj
});
...
// Add additional code
```

## http.post.promise(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTP POST request asynchronously.
	<p><b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">http.post(options)</a>. For more information on promises, see <a href="#">Promise Object</a>.</p>
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">http.post(options)</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/http Module</a>
<b>Since</b>	2015.2

## Syntax

**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
http.post.promise({
    url: 'http://www.google.com',
    body: 'My POST Data',
    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.put(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTP PUT request.
	<p><b>Important:</b> 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.</p>
<b>Returns</b>	<a href="#">http.ClientResponse</a>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/http Module</a>
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.url	string	required	The HTTP URL being requested	2015.2
options.body	string   Object	required	The PUT data.	2015.2
options.headers	Object	optional	The HTTP headers.  For more information, see <a href="#">HTTP Header Information</a> .	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing.

## Syntax

 <b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/http Module Script Samples</a> .
--

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
var response = http.put({
    url: 'http://www.google.com',
    body: 'My PUT Data',
    headers: headerObj
});
...
// Add additional code
```

## http.put.promise(options)

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Sends an HTTP PUT request asynchronously.
	 <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">http.put(options)</a> . For additional information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object

<b>Synchronous Version</b>	<code>http.put(options)</code>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/http Module</a>
<b>Since</b>	2015.2

## Syntax

 **Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
http.put.promise({
    url: 'http://www.google.com',
    body: 'My PUT Data',
    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.request(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTP request.
	 <b>Important:</b> 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.
<b>Returns</b>	<code>http.ClientResponse</code>

<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/http Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.method	enum	required	The HTTP request method. Set using the <a href="#">http.Method</a> enum.   <b>Note:</b> If the method is <b>DELETE</b> , this body data is ignored.	2015.2
options.url	string	required	The HTTP URL being requested	2015.2
options.body	string   Object	optional	The POST data if the method is <b>POST</b> .	
options.headers	Object	optional	The HTTP headers.  For more information, see <a href="#">HTTP Header Information</a> .	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required 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/http Module Script Samples](#).

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
var response = http.request({
    method: http.Method.GET,
```

```

    url: 'http://www.google.com',
    body: 'My REQUEST Data',
    headers: headerObj
});
...
// Add additional code

```

## http.request.promise(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTP request asynchronously.
	<p><b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">http.request(options)</a>. For more information on promises, see <a href="#">Promise Object</a>.</p>
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">http.request(options)</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	10 units
<b>Module</b>	N/http Module
<b>Since</b>	2015.2

### Syntax

**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```

// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
http.request.promise({
    method: http.Method.GET,
    url: 'http://www.google.com',
    body: 'My REQUEST Data',
    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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	<p>Holds the string values for supported cache durations. This enum is used to set the value of the <a href="#">ServerResponse.setCdnCacheable(options)</a> property.</p> <p><b>i Note:</b> JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
<b>Supported Script Types</b>	<p>Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Module</b>	<a href="#">N/http Module</a>

### Values

- LONG
- MEDIUM
- SHORT
- UNIQUE

### Syntax



**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 Samples](#).

```

// Add additional code
...
ServerResponse.setCdnCacheable({
    type: http.CacheDuration.MEDIUM
});
...
// Add additional code

```

## http.Method



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	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> .
	<p> <b>Note:</b> JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/http Module</a>

### 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/http Module Script Samples](#).

```
// Add additional code
...
var response = http.request({
    method: http.Method.GET,
    url: 'http://www.google.com'
});
...
// Add additional code
```

## http.RedirectType



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	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> .
-------------------------	---

	 <b>Note:</b> JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/http Module

## Values

- MEDIA\_ITEM
- RECORD
- RESTLET
- SUITELET
- TASK\_LINK

## Syntax

 **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 Samples](#).

```
// Add additional code
...
myServerResponseObj.sendRedirect({
  type: http.RedirectType.RECORD,
  identifier: record.Type.SALES_ORDER,
  parameters: {entity: 6}
});...
// Add additional code
```

# N/https Module

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

Load the N/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 scripts.

SecureString functionality is supported only in server scripts. You can also use this functionality to perform various string transformations using methods that hash, encode, or append another string.

You can use the N/https module to encode binary content or access a handle to the value in a NetSuite credential field.

When the N/https module is used, SuiteScript also loads the [N/crypto Module](#) and [N/encode Module](#).



**Important:** Use TLS 1.2 for HTTPS requests. SuiteScript 2.0 requests such `https.delete(options)`, `https.get(options)`, `https.post(options)`, `https.put(options)`, and `https.request(options)` usually go to third-party servers. Management of these servers is not within the control of your company. These HTTPS requests now fail the handshake when they attempt to connect to servers that do not support TLS 1.2. We recommend that you communicate with those who manage any third-party servers to which you connect, and ensure their servers support the TLS 1.2 protocol.



**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>

- [HTTPS Header Information](#).
- [N/https Module Members](#)
- [SecureString Object Members](#)
- [ClientResponse Object Members](#)
- [ServerResponse Object Members](#)
- [ServerRequest Object Members](#)
- [N/https Module Script Sample](#)

## HTTPS Header Information

HTTPS headers can be used to pass additional information with an HTTPS request or response. Each HTTPS header consists of its case-insensitive name followed by a colon (:), then by its value (without line breaks). For a general list of all HTTP headers (also applicable to HTTPS), visit <http://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>.

HTTPS headers are accessible in `https.ClientResponse`, `https.ServerRequest`, and `https.ServerResponse`.

In NetSuite, some headers are not supported. These are listed below as either general HTTPS headers or Suitelet response headers.

## General HTTPS Header Blacklist

Be aware that certain headers cannot be set manually when using N/https module methods. If a script attempts to set values for any of the following headers, the values are discarded. These headers are listed in the following table.

<ul style="list-style-type: none"> <li>■ Connection</li> <li>■ Content-Length</li> <li>■ Host</li> <li>■ JSESSIONID</li> <li>■ Trailer</li> </ul>	<ul style="list-style-type: none"> <li>■ Transfer-Encoding</li> <li>■ Upgrade</li> <li>■ Via</li> </ul>
---	---

## Suitelet Response HTTPS Header Blacklist

In addition to the headers described in [General HTTP 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 listed in the following table.

<ul style="list-style-type: none"> <li>■ Access-Control-Allow-Origin</li> <li>■ Allow</li> <li>■ Connection</li> <li>■ Content-Length</li> <li>■ Content-Location</li> <li>■ Content-MD5</li> <li>■ Content-Range</li> </ul>	<ul style="list-style-type: none"> <li>■ Date</li> <li>■ JSESSIONID</li> <li>■ Location</li> <li>■ Proxy-Authenticate</li> <li>■ Retry-After</li> <li>■ Server</li> <li>■ Trailer</li> <li>■ Via</li> </ul>	<ul style="list-style-type: none"> <li>■ Warning</li> <li>■ WWW-Authenticate</li> </ul>
--	---	---

## N/https Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">https.SecureString</a>	Object	Server scripts	Encapsulates data that may be sent to a third-party via an HTTPS call.
	<a href="#">https.ClientResponse</a>	Object (read-only)	Server scripts	Encapsulates the response to an HTTPS client request.
	<a href="#">https.ServerRequest</a>	Object (read-only)	Server scripts	Encapsulates the HTTPS request information sent to an HTTPS server. For example, a request received by a Suitelet or RESTlet.
	<a href="#">https.ServerResponse</a>	Object	Server scripts	Encapsulates the response from an HTTPS server to an HTTPS request. For example, a response from a Suitelet or RESTlet.
Method	<a href="#">https.createSecureKey(options)</a>	Object	Server scripts	Creates a key for the contents of a credential field.
	<a href="#">https.createSecureKey.promise(options)</a>	Object	Client scripts	Creates a key asynchronously for the contents of a credential field.
	<a href="#">https.createSecureString(options)</a>	Object	Server scripts	Creates an <a href="#">https.SecureString</a> Object.
	<a href="#">https.createSecureString.promise(options)</a>	Object	Client scripts	Creates an <a href="#">https.SecureString</a> Object asynchronously.
	<a href="#">https.delete(options)</a>	<a href="#">https.ClientResponse</a>	Server scripts	Sends an HTTPS DELETE request and returns the response.
	<a href="#">https.delete.promise(options)</a>	<a href="#">https.ClientResponse</a>	Client scripts	Sends an HTTPS DELETE request asynchronously and returns the response.
	<a href="#">https.get(options)</a>	<a href="#">https.ClientResponse</a>	Server scripts	Sends an HTTPS GET request and returns the response.
	<a href="#">https.get.promise(options)</a>	<a href="#">https.ClientResponse</a>	Client scripts	Sends an HTTPS GET request asynchronously and returns the response.
	<a href="#">https.post(options)</a>	<a href="#">https.ClientResponse</a>	Server scripts	Sends an HTTPS POST request and returns the response.
	<a href="#">https.post.promise(options)</a>	<a href="#">https.ClientResponse</a>	Client scripts	Sends an HTTPS POST request asynchronously and returns the response.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	<a href="#">https.put(options)</a>	<a href="#">https.ClientResponse</a>	Server scripts	Sends an HTTPS PUT request and returns the response.
	<a href="#">https.put.promise(options)</a>	<a href="#">https.ClientResponse</a>	Client scripts	Sends an HTTPS PUT asynchronously request and returns the response.
	<a href="#">https.request(options)</a>	<a href="#">https.ClientResponse</a>	Server scripts	Sends an HTTPS request and returns the response.  If a request fails, an error.SuiteScriptError is thrown.
	<a href="#">https.request.promise(options)</a>	<a href="#">https.ClientResponse</a>	Client scripts	Sends an HTTPS request asynchronously and returns the response.  If a request fails, a Promise.reject is thrown with a parameter Error.
Enum	<a href="#">https.CacheDuration</a>	enum	Server scripts	Holds the string values for supported cache durations. This enum is used to set the value of the <a href="#">ServerResponse.setCdnCacheable(options)</a> property.
	<a href="#">https.Encoding</a>	enum	Server scripts	Holds the string values for supported encoding types. This enum is used to set the value of parameters in <a href="#">SecureString.appendString(options)</a> , <a href="#">SecureString.convertEncoding(options)</a> , <a href="#">https.createSecureString(options)</a> .
	<a href="#">https.HashAlg</a>	enum	Server scripts	Holds the string values for supported hashing algorithms. This enum is used to set the value of parameters in <a href="#">SecureString.hash(options)</a> and <a href="#">SecureString.hmac(options)</a> .
	<a href="#">https.Method</a>	enum	Server scripts	Holds the string values for supported HTTP requests. This enum is used to set the value of parameters in <a href="#">https.request(options)</a> and to set the value of <a href="#">ServerRequest.method</a> .
	<a href="#">https.RedirectType</a>	enum	Server scripts	Holds the string values for supported NetSuite resources to which you can redirect. This enum is used to set the value of parameters in <a href="#">ServerResponse.sendRedirect(options)</a> .

## SecureString Object Members

The following members are called on the [https.SecureString](#) Object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">SecureString.appendSecureString(options)</a>	<a href="#">https.SecureString</a>	Server scripts	Appends a passed in <a href="#">https.SecureString</a> to another <a href="#">https.SecureString</a> .
	<a href="#">SecureString.appendString(options)</a>	<a href="#">https.SecureString</a>	Server scripts	Appends a passed in string to a <a href="#">https.SecureString</a> .
	<a href="#">SecureString.convertEncoding(options)</a>	<a href="#">https.SecureString</a>	Server scripts	Changes the encoding of a <a href="#">https.SecureString</a> .
	<a href="#">SecureString.hash(options)</a>	<a href="#">https.SecureString</a>	Server scripts	Produces the <a href="#">https.SecureString</a> as a hash.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	SecureString.hmac(options)	https.SecureString	Server scripts	Produces the https.SecureString as an hmac.

## ClientResponse Object Members

The following members are called on the [http.ClientResponse](#) Object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	ClientResponse.body	string (read-only)	Server scripts	The response body.
	ClientResponse.code	number (read-only)	Server scripts	The response code.
	ClientResponse.headers	Object (read-only)	Server scripts	The response body.

## ServerRequest Object Members

The following members are called on the [http.ServerRequest](#) Object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	ServerRequest.getLineCount(options)	number	Server scripts	Returns the number of lines in a sublist.
	ServerRequest.getSublistValue(options)	string	Server scripts	Returns the value of a sublist line item.
Property	ServerRequest.body	string (read-only)	Server scripts	The server request body
	ServerRequest.files	Object (read-only)	Server scripts	The server request files.
	ServerRequest.headers	Object (read-only)	Server scripts	The server request headers.
	ServerRequest.method	https.Method enum	Server scripts	The HTTPS method for the server request.
	ServerRequest.parameters	Object (read-only)	Server scripts	The server request parameters.
	ServerRequest.url	string (read-only)	Server scripts	The server request URL.

## ServerResponse Object Members

The following members are called on the [http.ServerResponse](#) Object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	ServerResponse.addHeader(options)	void	Server scripts	Adds a header to the response
	ServerResponse.getHeader(options)	string   string[]	Server scripts	Returns the value of a response header

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	ServerResponse.renderPdf(options)	void	Server scripts	Generates and renders a PDF directly to the response.
	ServerResponse.sendRedirect(options)	void	Server scripts	Sets the redirect URL by resolving to a NetSuite resource.
	ServerResponse.setCdnCacheable(options)	void	Server scripts	Sets CDN caching for a period of time.
	ServerResponse.setHeader(options)	void	Server scripts	Sets the value of a response header.
	ServerResponse.write(options)	void	Server scripts	Writes information (text/xml/html) to the response.
	ServerResponse.writeFile(options)	void	Server scripts	Writes a file to the response.
	ServerResponse.writeLine(options)	void	Server scripts	Writes line information (text/xml/html) to the response.
	ServerResponse.writePage(options)	void	Server scripts	Generates a page.
Property	ServerResponse.headers	Object (read-only)	Server scripts	The server response headers.

## N/https Module Script Sample

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to use a GUID to generate a secure token and a secret key. Note this sample 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 next sample (which is a Suitelet) for a more complete usage. To run this sample in the debugger, you must replace the GUID with one specific to your account.

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

require(['N/https', 'N/crypto', 'N/runtime'], function(http, https, runtime) {
    function createSecureString() {
        var passwordGuid = '{284CFB2D225B1D76FB94D150207E49DF}';
        var secureToken = https.createSecureString({
            input: passwordGuid
        });
        var secretKey = https.createSecretKey({
            input: passwordGuid
        });
    }
});
```

```

        secureToken = secureToken.hmac({
            algorithm: crypto.HashAlg.SHA256,
            key: secretKey
        });
    }
    createSecureString();
});

```

**i Note:** This script sample uses the `define` function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the `require` function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample shows how to create a form field that generates a GUID using a Suitelet. For more information about credential fields, see [Form.addCredentialField\(options\)](#).

**i 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.

```

/**
 * @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: ['<accountID>.app.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
};
});

```

## https.SecureString

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates data that may be sent to a third-party via an HTTPS call, such as a fragment of sensitive data.  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 <a href="#">SecureString Object Members</a> .
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	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 = '{284CFB2D225B1D76FB94D150207E49DF}';
    var secureToken = https.createSecureString({
        input: passwordGuid
    });
}

```

```

    });
}

...
// Add additional code

```

## SecureString.appendSecureString(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Appends the passed in <a href="#">https.SecureString</a> to another <a href="#">https.SecureString</a> .
<b>Returns</b>	<a href="#">https.SecureString</a>
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
options.secureString	<a href="#">https.SecureString</a>	required	The <a href="#">https.SecureString</a> to append.

### Syntax

 **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
...
string1.appendSecureString({
    secureString: secureString2
});
...
// Add additional code

```

## SecureString.appendString(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Appends the passed string to an <a href="#">https.SecureString</a> .
<b>Returns</b>	<a href="#">https.SecureString</a>
<b>Supported Script Types</b>	Server scripts

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/https Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.input	string	required	The string to append.
options.inputEncoding	<a href="#">https.Encoding</a>	required	The encoding of the string that is being appended.

## Syntax

 **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
...
string1.appendString({
    input: '48656c6c6f20776f726c640d0a',
    encoding: https.Encoding.HEX});
...
// Add additional code
```

## SecureString.convertEncoding(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Changes the encoding of a <a href="#">https.SecureString</a>
<b>Returns</b>	<a href="#">https.SecureString</a>
<b>Governance</b>	None
<b>Module</b>	N/https Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.toEncoding	<a href="#">https.Encoding</a>	required	The encoding to apply to the returned string.

## Syntax



**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 ...
https.convertEncoding({
    toEncoding: https.Encoding.HEX
});
...
// Add additional code
```

## SecureString.hash(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Hashes an https.SecureString Object
<b>Returns</b>	https.SecureString
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Parameters



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

Parameter	Type	Required / Optional	Description
options.algorithm	<a href="#">https.HashAlg</a>	required	The hash algorithm. Set the value using the <a href="#">https.HashAlg</a> enum.

## Syntax



**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
...
secureString = secureString.hash({
    algorithm: crypto.HashAlg.SHA256
});
...
// Add additional code
```

## SecureString.hmac(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Produces the <code>securestring</code> as an hmac.
<b>Returns</b>	<code>https.SecureString</code>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.algorithm</code>	<a href="#">https.HashAlg</a>	required	The hash algorithm. Set by the <a href="#">https.HashAlg</a> enum.
<code>options.key</code>	<a href="#">crypto.SecretKey</a>	required	A key returned from <a href="#">https.createSecureKey(options)</a> .

### Syntax

**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
...
secureToken = secureToken.hmac({
    algorithm: crypto.HashAlg.SHA256,
    key: secretKey
});
...
// Add additional code
```

## https.createSecureKey(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates and returns a <code>crypto.SecretKey</code> Object. This method can take a GUID. Use <code>Form.addCredentialField(options)</code> to generate a value.  You can put the key in your secure string. SuiteScript decrypts the value (key) and sends it to the server
---------------------------	---

<b>Returns</b>	<code>crypto.SecretKey</code>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.encoding</code>	<a href="#">https.Encoding</a>	optional	Specifies the encoding for the SecureKey.	2015.2
<code>options.guid</code>	string	required	A GUID used to generate a secret key.  The GUID can resolve to either data or metadata.	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
...
var secretKey = https.createSecureKey({
  encoding: https.Encoding.HEX,
  guid: '284CFB2D225B1D76FB94D150207E49DF'
});
...
// Add additional code
```

## https.createSecureKey.promise(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates and returns a <code>crypto.SecretKey</code> Object asynchronously.
	 <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">https.createSecureKey(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .

<b>Synchronous Version</b>	<a href="#">https.createSecureKey(options)</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Syntax

**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

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

...
// Add additional code
```

## https.createSecureString(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates and returns an <a href="#">https.SecureString</a> .
<b>Returns</b>	<a href="#">https.SecureString</a>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.input	string	required	The string to convert to a securestring.	Release 15 Version 2

Parameter	Type	Required / Optional	Description	Since
options. inputEncoding	https.Encoding	optional	Identifies the encoding that the input string uses.  The default value is <b>UTF_8</b>	Release 15 Version 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
...
var secureToken = https.createSecureString({
    input: passwordGuid
});
...
// Add additional code
```

## https.createSecureString.promise(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates and returns an <a href="#">https.SecureString</a> asynchronously.
	 <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">https.createSecureString(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">https.createSecureString(options)</a>
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Syntax

 **Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// Add additional code
...
var secureToken = https.createSecureString.promise({
```

```

    input: passwordGuid
});
...
// Add additional code

```

## https.ClientResponse



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the response to an HTTPS client request.  This object is read-only.  For a complete list of this object's properties, see <a href="#">ClientResponse Object Members</a> .
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	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
...
var clientResponse = https.get({
    url: 'https://www.testwebsite.com'
});
...
// Add additional code

```

## ClientResponse.body



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The client response body.  This property is read-only.
<b>Type</b>	string
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax

**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
...
var response = https.get({
    url: 'https://www.testwebsite.com'
});
log.debug({
    title: 'Client Response Body',
    details: response.body
});
...
// Add additional code
```

## ClientResponse.code

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The client HTTP response or status code.  This property is read-only.
<b>Type</b>	number
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax

**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
```

```

...
var response = https.get({
    url: 'https://www.testwebsite.com'
});
log.debug({
    title: 'Client Response Code',
    details: response.code
});
...
// Add additional code

```

## ClientResponse.headers

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The response header or headers.  This property is read-only.  For more information, see <a href="#">HTTPS Header Information</a> .
<b>Type</b>	Object
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

### Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

### Syntax

 **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
...
var response = https.get({
    url: 'https://www.testwebsite.com'
});
log.debug({
    title: 'Client Response Header',
    details: response.headers
});
...

```

```
// Add additional code
```

## https.ServerRequest



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the incoming HTTPS request information for an HTTPS 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 <a href="#">ServerRequest Object Members</a> .
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	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
...
serverRequest.getLineCount({
    group: 'sublistId'
});
...
// Add additional code
```

## ServerRequest.getLineCount(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the number of lines in a sublist.
<b>Returns</b>	number
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.group	string	required	The sublist internal ID.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required 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/https Module Script Sample](#).

```
// Add additional code
...
serverRequest.getLineCount({
    group: 'sublistId'
});
...
// Add additional code
```

## ServerRequest.getSublistValue(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the value of a sublist line item.
<b>Returns</b>	string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.group	string	required	The sublist internal ID.	2015.2
options.name	string	required	The name of the field.	2015.2

Parameter	Type	Required / Optional	Description	Since
options.line	string	required	The sublist line number.  <b>Note:</b> Sublist index starts at 0.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required 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/https Module Script Sample](#).

```
// Add additional code
...
serverRequest.getSublistValue({
  group: 'item',
  name: 'amount',
  line: '2'
});
...
// Add additional code
```

## ServerRequest.body

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The server request body. This property is read-only.
<b>Type</b>	string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax



**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
...
log.debug({
    title: 'Server Request Body',
    details: request.body
});
...
// Add additional code
```

## ServerRequest.files



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The server request files. This property is read-only.
<b>Type</b>	Object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax



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

```
// Add additional code
...
log.debug({
    title: 'Server Request Files',
    details: request.files
});
...
// Add additional code
```

```
var file = request.files['file_id'];
```

## ServerRequest.headers

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>This object represents a series of key/value pairs. Each pair represents a server request header name and its value.</p> <p>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).</p> <p>This property is read-only.</p> <p> <b>Important:</b> 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.</p>
	For more information, see <a href="#">HTTPS Header Information</a> .
<b>Type</b>	Object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

### Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

### Syntax

 **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
...
log.debug({
    title: 'Server Request Headers',
    details: request.headers
});
...
// Add additional code
```

## ServerRequest.method

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The server request HTTPS method. This property is read-only.
<b>Type</b>	enum
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

### Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

### Syntax

**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
...
log.debug({
    title: 'Server Request Method',
    details: request.method
});
...
// Add additional code
```

## ServerRequest.parameters

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The server request parameters. This property is read-only.
<b>Type</b>	Object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax

**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
...
log.debug({
    title: 'Server Request Parameters',
    details: request.parameters
});
...
// Add additional code
```

## ServerRequest.url

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The server request URL. This property is read-only.
<b>Type</b>	string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

## Syntax

**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
...
log.debug({
    title: 'Server Request URL',
```

```

    details: request.url
});
...
// Add additional code

```

## https.ServerResponse

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the response from an HTTPS server to an HTTPS request. For example, a response from a Suitelet or RESTlet.  For a complete list of this object's methods and properties, see <a href="#">ServerResponse Object Members</a> .
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	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
...
serverResponse.addHeader({
    name: 'Accept-Language',
    value: 'en-us',
});
...
// Add additional code

```

## ServerResponse.addHeader(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a header to the response.  If the same header has already been set, this method adds another line for that header. For example:
	<pre>{Vary: ['Accept-Language', 'Accept-Encoding']}</pre>
	For more information, see <a href="#">HTTPS Header Information</a> .
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/https Module
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	The name of the header.	2015.2
options.value	string	required	The value used to set the header.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing.
SSS_INVALID_HEADER	One or more headers are not valid.	The header name or value is invalid.

## Syntax

**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.addHeader({
    name: 'Accept-Language',
    value: 'en-us',
});
...
// Add additional code
```

## ServerResponse.getHeader(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the value or values of a response header. If multiple values are assigned to the header name, the values are returned as an Array.  For more information, see <a href="#">HTTPS Header Information</a> .
<b>Returns</b>	string   string[]
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None

<b>Module</b>	N/https Module
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	The name of the header.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required 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/https Module Script Sample](#).

```
// Add additional code
...
serverResponse.getHeader({
    name: 'Accept-Language'
});
...
// Add additional code
```

## ServerResponse.renderPdf(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Generates and renders a PDF directly to the response.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	N/https Module
<b>Since</b>	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 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/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.sendRedirect(options)

**ℹ️ Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates 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.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Parameters

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

**⚠️ Important:** All parameters must be prefixed with `custparam`.

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The type of resource to which the script redirects. Use the <a href="#">https.RedirectType</a> enum to set a value for this parameter.	2015.2
options.identifier	number   string	required	The primary ID for this resource. The value you use varies depending on the value of options.type, as follows:	2015.2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> <li>■ MEDIA_ITEM — Use the internal ID of a file stored in the NetSuite File Cabinet.</li> <li>■ RECORD — Use the <a href="#">record.Type</a> enum to identify the appropriate record type.</li> <li>■ RESTLET — Use the script ID from the script record of the appropriate RESTlet.</li> <li>■ SUITELET — Use the script ID from the script record of the appropriate Suitelet.</li> <li>■ TASK_LINK — Use the appropriate Task ID. Supported IDs are listed in <a href="#">Task IDs</a>.</li> </ul>	
options.id	string	optional	The secondary ID for this resource. If the options.type parameter is set to SUITELET or RESTLET, use the deployment ID. If the options.type parameter is set to RECORD, you can use the internal ID of a specific record instance.	2015.2
options.editMode	boolean <code>true</code>   <code>false</code>	optional	<p>Applicable when redirecting to a record resource.</p> <p>Specifies whether to return a URL for a record in edit mode or view mode.</p> <p>If set to <code>true</code>, returns the record in edit mode. If set to <code>false</code>, returns the record in view mode.</p> <p>The default value is <code>false</code>.</p>	2015.2
options.parameters	object	optional	Additional URL parameters as key-value pairs.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing. Note that this error is thrown if an enum is misspelled within a script. For example, you see this error if you use <code>http.RedirectType.TASKLINK</code> instead of <code>http.RedirectType.TASK_LINK</code> in the options.type field.
SSS_INVALID_URL_CATEGORY	The options.type: {type} is not valid. Please use the <a href="#">https.RedirectType</a> enum for supported types.	The script uses an unrecognizable string value for the options.type parameter. To avoid this error, use the <a href="#">https.RedirectType</a> enum.
INVALID_TASK_ID	The task ID: {id} is not valid. Please refer to the documentation for a list of supported task IDs.	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 <a href="#">Task IDs</a> .
INVALID_RCRD_TYPE	The record type {type} is invalid.	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 <a href="#">record.Type</a> enum to identify the appropriate record type.
INVALID_ID	You have provided an invalid script id or internal id: {id}	The options.type parameter is set to RESTLET or SUITELET, and the script uses an invalid ID 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/https Module Script Sample](#).

```
// Add additional code
...
myServerResponseObj.sendRedirect({
  type: https.RedirectType.RECORD,
  identifier: record.Type.SALES_ORDER,
  parameters: {entity: 8}
});
...
// Add additional code
```

## ServerResponse.setCdnCacheable(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets CDN caching for a period of time.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
options.type	<a href="#">https.CacheDuration</a> .	required	The value of the caching duration.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required 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/https Module Script Sample](#).

```
// Add additional code
```

```

...
serverResponse.setCdnCacheable({
    type: https.CacheDuration.LONG
});
...
// Add additional code

```

## ServerResponse.setHeader(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value of a response header.  For more information, see <a href="#">HTTPS Header Information</a> .
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

### Parameters

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	The name of the header.	2015.2
options.value	string	required	The value used to set the header.	2015.2

### Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing.
SSS_INVALID_HEADER	One or more headers are not valid.	The header name or value is invalid.

### Syntax



**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.setHeader({

```

```

    name: 'Accept-Language',
    value: 'en-us',
});
...
// Add additional code

```

## ServerResponse.write(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Writes information (text, xml, html) to the response.   <b>Note:</b> This method accepts only strings. To pass in a file, you can use <a href="#">ServerResponse.writeFile(options)</a> .
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

### Parameters

Parameter	Type	Required / Optional	Description	Since
options.output	string	required	The string being written.	2015.2

### Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing.
WRONG_PARAMETER_TYPE	{param name}	The value input for options.output is not a string.

### Syntax

 **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.write({
    output: 'Hello World'
});
...
// Add additional code

```

## ServerResponse.writeFile(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Writes a file to the response.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

### Parameters

Parameter	Type	Required / Optional	Description	Since
options.file	<a href="#">file.File</a>	required	A <a href="#">file.File</a> Object that encapsulates the file to be written.	2015.2
options.isInline	boolean <code>true</code>   <code>false</code>	optional	Determines whether the field is inline. If true, the file is inline.	2015.2

### Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing.
WRONG_PARAMETER_TYPE	{param name}	The value input for options.file is not a <a href="#">file.File</a> 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/https Module Script Sample](#).

```

// Add additional code
...
serverResponse.writeFile({

```

```

        file: myFileObj,
        isInline: true
    });
    ...
// Add additional code

```

## ServerResponse.writeLine(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Writes line information (text, xml, html) to the response.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

### Parameters

Parameter	Type	Required / Optional	Description	Since
options.output	string	required	The string being written.	2015.2

### Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing.
WRONG_PARAMETER_TYPE	{param name}	The value input for options.output is not a string.

### Syntax

 **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.writeLine({
    output: 'this is a sample string'
});
...

```

```
// Add additional code
```

## ServerResponse.writePage(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Generates a page.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

### Parameters

Parameter	Type	Required / Optional	Description	Since
options.pageObject	<a href="#">serverWidget.Assistant</a>   <a href="#">serverWidget.Form</a>   <a href="#">serverWidget.List</a>	required	A standalone page Object in the form of an assistant, form or list.	2015.2

### Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required 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/https Module Script Sample](#).

```
// Add additional code
...
var myPageObj = serverWidget.createList({
  title: 'Simple List'
});

ServerResponse.writePage({
  pageObject: myPageObj
});
...
// Add additional code
```

## ServerResponse.headers

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The server response headers.  This property is read-only.  For more information, see <a href="#">HTTPS Header Information</a> .
<b>Type</b>	Object  Note that If multiple values are assigned to one header name, the values are returned as an array. For example:
	<pre>{Vary: ['Accept-Language', 'Accept-Encoding']}</pre>
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

### Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

### Syntax

 **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
...
log.debug({
    title: "Server Response Headers",
    details: serverResponse.headers
});
...
// Add additional code
```

## https.get(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTPS GET request.
<b>Returns</b>	<a href="#">https.ClientResponse</a>

<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.url	string	required	The HTTPS URL being requested	2015.2
options.headers	Object	optional	The HTTPS headers.  For more information, see <a href="#">HTTPS Header Information</a> .	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required 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/https Module Script Sample](#).

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
var response = https.get({
    url: 'https://www.testwebsite.com',
    headers: headerObj
});
...
// Add additional code
```

## https.get.promise(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTPS GET request asynchronously.
---------------------------	--

	<b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">https.get(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">https.get(options)</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	10 units
<b>Module</b>	N/https Module
<b>Since</b>	2015.2

## Syntax

**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
https.get.promise({
    url: 'https://www.testwebsite.com',
    headers: headerObj
})
    .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)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTPS DELETE request.
---------------------------	--------------------------------

	<p><b>Important:</b> 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.</p>
	<p><b>Note:</b> This method does not include an <code>options.body</code> parameter. Postdata is not required when the HTTPS method is a DELETE request.</p>
<b>Returns</b>	<code>https.ClientResponse</code>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
<code>options.url</code>	string	required	The HTTPS URL being requested	2015.2
<code>options.headers</code>	Object	optional	The HTTPS headers.  For more information, see <a href="#">HTTPS Header Information</a> .	2015.2

## Errors

Error Code	Message	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	Missing a required argument: {param name}	A required parameter is missing.

## Syntax

<b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/https Module Script Sample</a> .
--

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
var response = https.delete({
    url: 'https://www.mytestwebsite.com',
    headers: headerObj
});
...
```

```
// Add additional code
```

## https.delete.promise(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTP DELETE request asynchronously.   <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">https.delete(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">https.delete(options)</a>
<b>Supported Script Types</b>	Client scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

### Syntax

 **Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
https.delete.promise({
    url: 'https://www.mytestwebsite.com',
    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.post(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTPS POST request.
	<b>Important:</b> 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.
<b>Returns</b>	<a href="#">https.ClientResponse</a>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.url	string	required	The HTTPS URL being requested	2015.2
options.body	string   Object	required	The POST data.	
options.headers	Object	optional	The HTTPS headers. For more information, see <a href="#">HTTPS Header Information</a> .	2015.2

### Errors

Error Code	Message	Thrown If
SSS_INVALID_URL	The URL must be a fully qualified HTTP/HTTPS URL	An incorrect protocol is used, such as using HTTP within the HTTPS module.
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is missing.
SSS_REQUEST_LOOP_DETECTED	This script executes a recursive function that has exceeded the limit for the number of times a script can call itself using an HTTP request.	A script is calling back into itself recursively via an HTTP/HTTPS request.

Error Code	Message	Thrown If
	Please examine the script for a potential infinite recursion problem.	

## Syntax

 **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
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};var response = https.post({
    url: 'https://www.testwebsite.com',
    body: 'My POST Data',
    headers: headerObj
});
...
// Add additional code
```

## https.post.promise(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTPS POST request asynchronously.
	 <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">https.post(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">https.post(options)</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

## Syntax

 **Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// Add additional code
```

```

...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};https.post.promise({
    url: 'https://www.testwebsite.com',
    body: 'My POST Data',
    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.put(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTPS PUT request.  <div style="border: 1px solid #fca; padding: 5px;"> <span style="color: yellow;"></span> <b>Important:</b> 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.         </div>
<b>Returns</b>	<a href="#">https.ClientResponse</a>
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.url	string	required	The HTTPS URL being requested	2015.2

Parameter	Type	Required / Optional	Description	Since
options.body	string   Object	required	The PUT data.	
options.headers	Object	optional	The HTTPS headers. For more information, see <a href="#">HTTPS Header Information</a> .	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required 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/https Module Script Sample](#).

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
var response = https.put({
    url: 'https://www.testwebsite.com',
    body: 'My PUT Data',
    headers: headerObj
});
...
// Add additional code
```

## https.put.promise(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTPS PUT request asynchronously.
	<p><b>i Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">https.put(options)</a>. For more information on promises, see <a href="#">Promise Object</a>.</p>
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">https.put(options)</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .

<b>Governance</b>	10 units
<b>Module</b>	N/https Module
<b>Since</b>	2015.2

## Syntax

 **Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
https.put.promise({
    url: 'https://www.testwebsite.com',
    body: 'My PUT Data',
    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.request(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTPS request.
 <b>Important:</b> 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.	
<b>Returns</b>	<a href="#">https.ClientResponse</a> .
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Governance</b>	10 units
<b>Module</b>	N/https Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.method	enum	required	The HTTPS request method. Set using the <a href="#">https.Method</a> enum.	2015.2
options.url	string	required	The HTTPS URL being requested	2015.2
options.body	string   Object	optional	The POST data if the method is <b>POST</b> .   <b>Note:</b> If the method is <b>DELETE</b> , this body data is ignored.	
options.headers	Object	optional	The HTTPS headers.  For more information, see <a href="#">HTTPS Header Information</a> .	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required 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/https Module Script Sample](#).

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
var response = https.request({
    method: https.Method.GET,
    url: 'https://www.testwebsite.com',
    body: 'My REQUEST Data',
    headers: headerObj
});
...
// Add additional code
```

## https.request.promise(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sends an HTTP request asynchronously.
	<b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">https.request(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">https.request(options)</a>
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/https Module</a>
<b>Since</b>	2015.2

### Syntax



**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// Add additional code
...
var headerObj = {
    name: 'Accept-Language',
    value: 'en-us'
};
https.request.promise({
    method: https.Method.GET,
    url: 'https://www.testwebsite.com',
    body: 'My REQUEST Data',
    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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the string values for supported cache durations. This enum is used to set the value of the <a href="#">ServerResponse.setCdnCacheable(options)</a> property.
	<b>Note:</b> JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>

### Values

- LONG
- MEDIUM
- SHORT
- UNIQUE

### Syntax



**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.setCdnCacheable({
  type: https.CacheDuration.LONG
});
...
// Add additional code
```

## https.Encoding



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the string values for supported encoding values.
-------------------------	--

	<p><b>Note:</b> JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/https Module

## 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/https Module Script Sample](#).

```
// Add additional code
...
var mySecretKey = https.createSecretKey({
    encoding: https.Encoding.HEX,
    guid: '284CFB2D225B1D76FB94D150207E49DF'
});
...
// Add additional code
```

## https.HashAlg

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the string values for supported hashing algorithms.
	<p><b>Note:</b> JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/https Module

## 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/https Module Script Sample](#).

```
// Add additional code
...
var mySecureString = https.createSecureString({
    input: 'ConvertMe'
});
var mySecureStringHash = mySecureString.hash({
    algorithm: https.HashAlg.SHA256
});
...
// Add additional code
```

## https.Method

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the string values for supported HTTPS requests. This enum is used to set the value of <code>https.request(options)</code> and <code>ServerRequest.method</code> .
	<p> <b>Note:</b> JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>

## 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](#).

```
// Add additional code
...
var response = https.request({
  method: https.Method.GET,
  url: 'https://www.testwebsite.com'
});
...
// Add additional code
```

## https.RedirectType



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	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 <a href="#">ServerResponse.sendRedirect(options)</a> .
	<p><b>Note:</b> JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/https Module</a>

## Values

Value	Description
MEDIA_ITEM	A file in the NetSuite File Cabinet
RECORD	A NetSuite record.
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](#).

```
// Add additional code
```

```
...
myServerResponseObj.sendRedirect({
    type: https.RedirectType.RECORD,
    identifier: record.Type.SALES_ORDER,
    parameters: {entity: 6}
});...
// Add additional code
```

## N/https/clientCertificate Module

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

Load the clientCertificate module to send SSL requests with a digital certificate.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Method	clientCertificate. post(options)	https.ClientResponse	Server-side scripts	Sends a SSL secured POST request to a remote server.
	clientCertificate. get(options)	https.ClientResponse	Server-side scripts	Sends a SSL secured GET request to a remote server.
	clientCertificate. put(options)	https.ClientResponse	Server-side scripts	Sends a SSL secured PUT request to a remote server.
	clientCertificate. delete(options)	https.ClientResponse	Server-side scripts	Sends a SSL secured DELETE request to a remote server.
	clientCertificate. request(options)	https.ClientResponse	Server-side scripts	Sends a SSL secured REQUEST request to a remote server.

## N/https/clientCertificate Module Script Sample

The following is an example of how to send a certificate to a Brazilian tax authority for authentication.

**i Note:** This sample script uses the **require** function so that you can copy it into the SuiteScript Debugger and test it. You must use the **define** function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample sends a secure post request to a remote URL.

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

```

require(['N/https/clientCertificate'],function (cert){
    var url = "https://nfe.fazenda.sp.gov.br/ws/cadconsultacadastro4.asmx";
    var data = "<?xml version='1.0' encoding='utf-8'?><soapenv:Envelope xmlns:soapenv='http://www.w3.org/2003/05/soap-envelope'><soapenv:Body><ns1:nfeDadosMsg xmlns:ns1='http://www.portalfiscal.inf.br/nfe/wsdl/CadConsultaCadastro4'><ConsCad xmlns='http://www.portalfiscal.inf.br/nfe' versao='2.00'><infCons><xServ>CONS-CAD</xServ><UF>SP</UF><CNPJ>47508411000156</CNPJ></infCons> </ConsCad></ns1:nfeDadosMsg></soapenv:Body></soapenv:Envelope>";
    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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to send a SSL secured POST request to a remote service and return the response.
	 <b>Important:</b> 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.
<b>Returns</b>	An <a href="#">https.ClientResponse</a> Object
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/https/clientCertificate Module</a>
<b>Since</b>	2019.1

## Parameters

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

Parameter	Type	Required/Optional	Description	Since
options.url	string	required	The URL address of the remote server.	2019.1

<code>options.certId</code>	string	required	The ID of the client certificate.	2019.1
<code>options.body</code>	string	required	The POST data to be sent to the remote server.	2019.1
<code>options.headers</code>	object	optional	The HTTPS headers associated with the request.	2019.1

## clientCertificate.get(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to send a SSL secured GET request to a remote service and return the response.
	 <b>Important:</b> 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.
<b>Returns</b>	An <a href="#">https.ClientResponse</a> Object
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/https/clientCertificate Module</a>
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required/Optional	Description	Since
<code>options.url</code>	string	required	The URL address of the remote server.	2019.2
<code>options.certId</code>	string	required	The ID of the client certificate.	2019.2
<code>options.headers</code>	object	optional	The HTTPS headers associated with the request.	2019.2

## clientCertificate.put(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to send a SSL secured request to a remote service and return the response.
---------------------------	--

	 <b>Important:</b> 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.
<b>Returns</b>	An <a href="#">https.ClientResponse</a> Object
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/https/clientCertificate Module</a>
<b>Since</b>	2019.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required/Optional	Description	Since
<code>options.url</code>	string	required	The URL address of the remote server.	2019.2
<code>options.body</code>	string	required	The PUT data to be sent to the remote server.	2019.2
<code>options.certId</code>	string	required	The ID of the client certificate.	2019.2
<code>options.headers</code>	object	optional	The HTTPS headers associated with the request.	2019.2

## clientCertificate.delete(options)

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Method used to send a SSL secured request to a remote service and return the response.
	 <b>Important:</b> 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.
<b>Returns</b>	An <a href="#">https.ClientResponse</a> Object
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/https/clientCertificate Module</a>
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required/Optional	Description	Since
<code>options.url</code>	string	required	The URL address of the remote server.	2019.2
<code>options.certId</code>	string	required	The ID of the client certificate.	2019.2
<code>options.headers</code>	object	optional	The HTTPS headers associated with the request.	2019.2

## clientCertificate.request(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to send a SSL secured request to a remote service and return the response.
	 <b>Important:</b> 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.
<b>Returns</b>	An <a href="#">https.ClientResponse</a> Object
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/https/clientCertificate Module</a>
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required/Optional	Description	Since
<code>options.url</code>	string	required	The URL address of the remote server.	2019.2
<code>options.body</code>	string	required for PUT and POST methods   optional for HEAD, GET, DELETE	The REQUEST data to be sent to the remote server.	2019.2
<code>options.certId</code>	string	required	The ID of the client certificate.	2019.2
<code>options.headers</code>	object	required	The HTTP headers associated with the request.	2019.2

<code>options.method</code>	string	required	The HTTP method to be used. Use the <a href="#">https.Method</a> enum to set this value.	2019.2
-----------------------------	--------	----------	--	--------

## N/keyControl Module



**Note:** The content in this help topic pertains to SuiteScript 2.0.

The N/keyControl module can access key storage, which is also available in the UI at Setup > Company > Preferences > Keys. By using the SSH keys, you can manage files and directories by using the SSH file transfer (SFTP) protocol. For more information, see the help topic [SSH Keys for SFTP](#).

For more information about SFTP, see [N/sftp Module](#).

### N/keyControl Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">keyControl.Key</a>	Object	Server-side scripts	Represents the key object.
Method	<a href="#">keyControl.findKeys(options)</a>	Object	Server-side scripts	Searches and returns a list of keys based on criteria set. If no options are set for criteria, the full list of keys stored in NetSuite is returned.
	<a href="#">keyControl.createKey(options)</a>	<a href="#">keyControl.Key</a>	Server-side scripts	Creates a key.
	<a href="#">keyControl.deleteKey(options)</a>	Object	Server-side scripts	Marks the key as deleted in database. The history is retained.
	<a href="#">keyControl.loadKey(options)</a>	Object	Server-side scripts	Loads a key.
Enum	<a href="#">keyControl.Operator</a>	enum	Server-side scripts	Holds the values for key operators.

### Key Object Members

The following members are called on the [keyControl.Key](#) object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	<a href="#">Key.file</a>	<a href="#">file.File</a>	Server-side scripts	File object of the key.
	<a href="#">Key.password</a>	Object	Server-side scripts	Password of the key (write-only). You can create a GUID using <a href="#">Form.addSecretKeyField(options)</a> .
	<a href="#">Key.scriptId</a>	Object	Server-side scripts	Script ID of the key.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	Key.name	Object	Server-side scripts	Name of the key.
	Key.description	Object	Server-side scripts	Description of the key.
	Key.restrictions	Object	Server-side scripts	The internal IDs of the employees selected in the <b>Restrict to Employees</b> field of the key record.
Method	Key.save()	Object	Server-side scripts	Saves the key.

## N/keyControl Module Script Sample

### Example 1

**i Note:** This sample script uses the **require** function so that you can copy it into the SuiteScript Debugger and test it. You must use the **define** function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how you can create a key.

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

require(['N/keyControl','N/file'],function(keyControl,file){
    var key = keyControl.createKey();
    key.file = file.load(422);
    //id of file containing private key (id_ecdsa or id_rsa)
    key.name = "SFTP key";
    key.save();
})
```

### Example 2

**i Note:** This script sample uses the **define** function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the **require** function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample shows how you can add a secret key field.

```
/**
 * @NApiVersion 2.0
 * @NScriptType Suitelet
 */

define(['N/ui/serverWidget', 'N/file', 'N/keyControl','N/runtime'],
    function(ui, file, keyControl, runtime) {
```

```

function onRequest(context) {
    var request = context.request;
    var response = context.response;
    if (request.method === 'GET') {
        var form = ui.createForm({
            title: 'Enter Password'
        });
        var credField = form.addSecretKeyField({
            id: 'custfield_password',
            label: 'Password',
            restrictToScriptIds: [runtime.getCurrentScript().id],
            restrictToCurrentUser: true //Depends on use case
        });
        credField.maxLength = 64;
        form.addSubmitButton();
        response.writePage(form);
    } else{
        // Read the request parameter matching the field ID we specified in the form
        var passwordToken = request.parameters.custfield_password;
        var pem = file.load({
            id:422
        });
        var key = keyControl.createKey();
        key.file = pem;
        key.name = 'Test';
        key.password = passwordToken;
        key.save();
    }
}
return {
    onRequest: onRequest
};
});

```

## keyControl.Key



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Represents the key.
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/keyControl Module</a>
<b>Methods and Properties</b>	<a href="#">Key Object Members</a>
<b>Since</b>	2019.2

## Syntax

```
// Add additional code
```

```

...
var key = keyControl.createKey();
    key.file = file.load(422);
    //id of file containing private key (id_ecdsa or id_rsa)
...
// Add additional code

```

## Key.file



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The file object of the key.
<b>Type</b>	file.File
<b>Module</b>	N/keyControl Module
<b>Parent Object</b>	keyControl.Key
<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Methods and Properties</b>	<a href="#">Key Object Members</a>
<b>Since</b>	2019.2

## 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/keyControl Module Script Sample](#).

```

// Add additional code
...
var key = keyControl.createKey();
key.file = file.load(422);
...
// Add additional code

```

## Key.password



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The password of the key. GUID or secret token for working with passwords is accepted.
<b>Type</b>	String (write-only)
<b>Module</b>	N/keyControl Module
<b>Parent Object</b>	keyControl.Key

<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Methods and Properties</b>	<a href="#">Key Object Members</a>
<b>Since</b>	2019.2

## 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/keyControl Module Script Sample](#).

```
// Add additional code
...
var passwordToken = request.parameters.custfield_password;
    var pem = file.load({id:422});
    var key = keyControl.createKey();
key.file = pem;
key.name = 'Test';
key.password = passwordToken;
...
// Add additional code
```

## Key.scriptId



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The script ID of the key. Using <a href="#">Key.save()</a> and <a href="#">keyControl.findKeys(options)</a> returns the script ID.
<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/keyControl Module</a>
<b>Methods and Properties</b>	<a href="#">Key Object Members</a>
<b>Since</b>	2019.2

## 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/keyControl Module Script Sample](#).

```
// Add additional code
...
var key = keyControl.createKey();
key.scriptId = 'testid'
...
```

```
// Add additional code
```

## Key.name



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The name of the key.
<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/keyControl Module</a>
<b>Methods and Properties</b>	<a href="#">Key Object Members</a>
<b>Since</b>	2019.2

## 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/keyControl Module Script Sample](#).

```
// Add additional code
...
var key = keyControl.createKey();
key.name = 'testname'
...
// Add additional code
```

## Key.description



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The description of the key.
<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/keyControl Module</a>
<b>Methods and Properties</b>	<a href="#">Key Object Members</a>
<b>Since</b>	2019.2

## 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/keyControl Module Script Sample](#).

```
// Add additional code
```

```
...
var key = keyControl.createKey();
key.description = 'testdescription'
...
// Add additional code
```

## Key.restrictions



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	An array of employee IDs. Only these employees can access the key.
<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/keyControl Module</a>
<b>Methods and Properties</b>	<a href="#">Key Object Members</a>
<b>Since</b>	2019.2

## 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/keyControl Module Script Sample](#).

```
// Add additional code
...
var key = keyControl.createKey();
key.restriction = 'testrestrictions'
...
// Add additional code
```

## Key.save()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Saves the key.
<b>Returns</b>	Object
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">Key Object Members</a>
<b>Since</b>	2019.2

## 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/keyControl Module Script Sample](#).

```
// Add additional code
...
var key = keyControl.createKey();
key.file = file.load(422);
//id of file containing private key
key.name = 'SFTP key';
key.save();
...
// Add additional code
```

## keyControl.createKey(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a key record on the <a href="#">Private Keys</a> page using a file from the File Cabinet.
<b>Returns</b>	<a href="#">keyControl.Key</a>
<b>Supported Script Types</b>	Server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/keyControl Module</a>
<b>Since</b>	2019.2

## Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.file	file	optional	The file with the key.	2019.2
options.password	string	optional	The password that is associated with the key.	2019.2
options.scriptId	string	optional	The script ID for the newly-created key.	2019.2
options.description	string	optional	The description of the key.	2019.2
options.restrictions	number[] or string[]	optional	The array of employee internal IDs selected in the <b>Restricted to Employees</b> field for a key.	2019.2

Parameter	Type	Required / Optional	Description	Since
options.name	string	optional	The name of the key.	2019.2

## 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/keyControl Module Script Sample](#).

```
// Add additional code
...
require(['N/keyControl','N/file'],function(keyControl,file){
    var key = keyControl.createKey();
    key.file = file.load(422);
    //id of file containing private key (id_ecdsa or id_rsa)
    key.name = "SFTP key";
    key.save();
})
...
// Add addtional code
```

## keyControl.findKeys(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a list of keys that are available to the user.
<b>Returns</b>	Metadata about the keys
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/keyControl Module</a>
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.restriction	number	optional	The internal ID of an employee selected in the <b>Restrict to Employees</b> field.	2019.2
options.name	string or object	optional	The name of the key. The properties of the object are:	2019.2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> <li>■ <b>value</b> is a string, which can be used if object is used instead of string.</li> <li>■ <b>operator</b> is one of the operator enum.</li> <li>■ <b>ignoreCase</b> is either true or false.</li> </ul> <p>If the object is used, the <b>value</b> is mandatory. <b>Operator</b> defaults to <b>equals</b> and <b>ignoreCase</b> defaults to true.</p>	
<b>options.description</b>	string or object	optional	<p>The description of the key.</p> <p>The properties of the object are:</p> <ul style="list-style-type: none"> <li>■ <b>value</b> is a string, which can be used if object is used instead of string.</li> <li>■ <b>operator</b> is one of the operator enum.</li> <li>■ <b>ignoreCase</b> is either true or false.</li> </ul> <p>If the object is used, the <b>value</b> is mandatory. <b>Operator</b> defaults to <b>equals</b> and <b>ignoreCase</b> defaults to true.</p>	2019.2

## 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/keyControl Module Script Sample](#).

```
// Add additional code
...
require(['N/keyControl'],function(keyControl){
  var keys = keyControl.findKeys({
    name:{value: 'test',
    operator: keyControl.Operator.CONTAINS, ignoreCase:true}
  });
  ...
// Add additional code
```

## keyControl.deleteKey(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Deletes a key.
<b>Returns</b>	Object
<b>Supported Script Types</b>	Server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/keyControl Module</a>
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.scriptId	string	required	The script ID of the key to be deleted.  Using <a href="#">Key.save()</a> and <a href="#">keyControl.findKeys(options)</a> returns the script ID.	2019.2

## 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/keyControl Module Script Sample](#).

```
// Add additional code
...
var keyId = keyControl.deleteKey({
   scriptId: 'key_test'
});;
...
// Add additional code
```

## keyControl.loadKey(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Loads a key.
<b>Returns</b>	Object
<b>Supported Script Types</b>	Server-side scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/keyControl Module</a>
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.scriptId	string	required	The script ID of the key to be loaded.  Using <a href="#">Key.save()</a> and <a href="#">keyControl.findKeys(options)</a> returns the script ID.	2019.2

## 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/keyControl Module Script Sample](#).

```
// Add additional code
...
var key = keyControl.loadKey({
   scriptId: '_testKey'
});
...
// Add additional code
```

## keyControl.Operator



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the values for the key operators of <a href="#">keyControl.findKeys(options)</a> .
	<p><b>Note:</b> 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>
<b>Module</b>	<a href="#">N/keyControl Module</a>
<b>Supported Script Types</b>	All server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Since</b>	2019.2

## Values

Value	Sets Property To
STARTS_WITH	<code>startswith</code>
CONTAINS	<code>contains</code>
ENDS_WITH	<code>endswith</code>
EQUALS	<code>equals</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/keyControl Module Script Sample](#).

```
require(['N/keyControl'],function(keyControl){
    var keys = keyControl.findKeys({
        name:{
            value: 'test',
```

```

    operator: keyControl.Operator.CONTAINS}
});

})

```

## N/log Module

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

The log methods for logging script execution details can be accessed globally or by loading the N/log module. Load the N/log module when you want to manually access its members, such as for testing purposes. For more information about the global log object, see [Log Object](#).

- [N/log Module Members](#)
- [N/log Module Guidelines](#)
- [Using Log Levels](#)
- [Viewing Script Execution Logs](#)
- [N/log Module Script Sample](#)
- [Governance on Script Logging](#)

### N/log Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">log.audit(options)</a>	void	Client and server scripts	Logs an Audit type log message to the Execution Log tab of the script deployment for the current script.
	<a href="#">log.debug(options)</a>	void	Client and server scripts	Logs a Debug type log message to the Execution Log tab of the script deployment for the current script.
	<a href="#">log.emergency(options)</a>	void	Client and server scripts	Logs an Emergency type log message to the Execution Log tab of the script deployment for the current script.
	<a href="#">log.error(options)</a>	void	Client and serverscripts	Logs an Error type log message to the Execution Log tab of the script deployment for the current script.

### N/log Module Guidelines

The following guidelines are provided for use of the N/log module:

- 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.

```

...
// 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:

Log Level	Description
Debug	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.
Audit	Shows a record of events that have occurred during the processing of the script (for example, "A request was made to an external site.").
Error	Shows only unexpected script errors.
Emergency	Shows only the most critical errors in the script log.

## 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 . 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.

When you debug a script in the SuiteScript Debugger, log details appear on the Execution Log tab of the SuiteScript Debugger. These details do not appear on the Script Deployment page for the script.

## N/log Module Script Sample

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to create debug log messages.

```
/**  
 * @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)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Logs an Audit type log message 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.
<b>Returns</b>	void
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	Amount of logging in any 60 minute period is limited. See <a href="#">N/log Module Guidelines</a> .
<b>Module</b>	<a href="#">N/log Module</a>

<b>Since</b>	2016.1
--------------	--------

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
title	string	optional	<p>String to appear in the Title column on the Execution Log tab of the script deployment.</p> <p>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.</p>	2016.1
details	any	required	<p>You can pass any value for this parameter.</p> <p>If the value is a JavaScript Object type, <a href="#">JSON.stringify(obj)</a> is called on the object before displaying the value.</p> <p>NetSuite truncates any resulting string over 3999 characters.</p>	2016.1

## Syntax

The following code snippet shows the syntax for this method.

```
//Add additional code
...
var var1 = 'value';
log.audit({
    title: 'Audit Entry',
    details: 'Value of var1 is: ' + var1
});
...
//Add additional code
```

## log.debug(options)

<b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.
--

<b>Method Description</b>	Logs a Debug type log message 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.
<b>Returns</b>	void
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Governance</b>	Amount of logging in any 60 minute period is limited. See <a href="#">N/log Module Guidelines</a> .
<b>Module</b>	<a href="#">N/log Module</a>
<b>Since</b>	2016.1

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
title	string	required	<p>String to appear in the Title column on the Execution Log tab of the script deployment.</p> <p>Maximum length is 99 characters.</p> <p>If you set this value to null, an empty string (""), or omit it, the word "Untitled" appears for the log entry.</p>	2016.1
details	any	optional	<p>You can pass any value for this parameter.</p> <p>If the value is a JavaScript object type, <a href="#">JSON.stringify(obj)</a> is called on the object before displaying the value.</p> <p>NetSuite truncates any resulting string over 3999 characters.</p>	2016.1

## Syntax

The following code snippet shows the syntax for this method.

```
//Add additional code
...
var var1 = 'value';
log.debug({
    title: 'Debug Entry',
    details: 'Value of var1 is: ' + var1
});
...
//Add additional code
```

## log.emergency(options)

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Logs an Emergency type log message to the Execution Log tab of the script deployment for the current script.  Use this method for scripts in production.
<b>Returns</b>	void
<b>Supported Script Types</b>	All script types

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	Amount of logging in any 60 minute period is limited. See <a href="#">N/log Module Guidelines</a> .
<b>Module</b>	<a href="#">N/log Module</a>
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
title	string	optional	<p>String to appear in the Title column on the Execution Log tab of the script deployment.</p> <p>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.</p>	2016.1
details	any	required	<p>You can pass any value for this parameter.</p> <p>If the value is a JavaScript Object type, <a href="#">JSON.stringify(obj)</a> is called on the object before displaying the value.</p> <p>NetSuite truncates any resulting string over 3999 characters.</p>	2016.1

## Syntax

The following code snippet shows the syntax for this method.

```
//Add additional code
...
var var1 = 'value';
log.emergency({
    title: 'Emergency Entry',
    details: 'Value of var1 is: ' + var1
});
...
//Add additional code
```

## log.error(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Logs an Error type log message to the Execution Log tab of the script deployment for the current script.</p> <p>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.</p> <p>Use this method for scripts in production.</p>
---------------------------	---

<b>Returns</b>	void
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	Amount of logging in any 60 minute period is limited. See <a href="#">N/log Module Guidelines</a> .
<b>Module</b>	<a href="#">N/log Module</a>
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
title	string	optional	<p>String to appear in the Title column on the Execution Log tab of the script deployment.</p> <p>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.</p>	2016.1
details	any	required	<p>You can pass any value for this parameter.</p> <p>If the value is a JavaScript object type, <a href="#">JSON.stringify(obj)</a> is called on the object before displaying the value.</p> <p>NetSuite truncates any resulting string over 3999 characters.</p>	2016.1

## Syntax

The following code snippet shows the syntax for this method.

```
//Add additional code
...
var var1 = 'value';
log.error({
    title: 'Error Entry',
    details: 'Value of var1 is: ' + var1
});
...
//Add additional code
```

## N/piremoval Module

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

Load the N/piremoval module to remove personal information (PI) from system notes, workflow history, and specific field values. Use the N/piremoval module to comply with the General Data Protection Regulation (GDPR), specifically the right to be forgotten. You can remove personal information from

system notes only, or you can also remove workflow history and field values on the record. Entity records, transactions, and custom records are supported.

You can use the [piremoval.createTask\(options\)](#) method to create a PI removal task, or use [piremoval.loadTask\(options\)](#) to load an existing PI removal task. Both of these methods return a [piremoval.PiRemovalTask](#) object that represents the task. Create a [piremoval.PiRemovalTask](#) object for each record type that requires removal of personal information. Use the [PiRemovalTask.save\(\)](#) method to save the task, then use the [PiRemovalTask.run\(\)](#) method to process the task and remove the personal information.

You can use the [piremoval.getTaskStatus\(options\)](#) method to check the status of a submitted PI removal task. This method returns a [piremoval.PiRemovalTaskStatus](#) object that describes the current status of the removal task. The [piremoval.PiRemovalTaskStatus](#) object uses an iterator to provide a list of log entries in the [PiRemovalTaskStatus.logList](#) object.

To use the N/piremoval module, the following requirements must be met:

- Remove Personal Information Create permission is required to create a PI removal task.
- Remove Personal Information Run permission is required to run a PI removal task.

For more information, see the help topic [Personal Information \(PI\) Removal](#).

### In this help topic

- [N/piremoval Module Members](#)
- [PiRemovalTask Object Members](#)
- [PiRemovalTaskLogItem Object Members](#)
- [PiRemovalTaskStatus Object Members](#)
- [N/piremoval Module Script Samples](#)

## N/piremoval Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">piremoval.PiRemovalTask</a>	Object	Server scripts	Encapsulates a personal information removal task. Use <a href="#">piremoval.createTask(options)</a> to create this object.
	<a href="#">piremoval.PiRemovalTaskLogItem</a>	Object	Server scripts	Encapsulates a log item of the personal information removal task status.
	<a href="#">piremoval.PiRemovalTaskStatus</a>	Object	Server scripts	Encapsulates the status of a personal information removal task. Use <a href="#">piremoval.getTaskStatus(options)</a> to create this object.
Method	<a href="#">piremoval.createTask(options)</a>	<a href="#">piremoval.PiRemovalTask</a>	Server scripts	Creates a personal information removal task.
	<a href="#">piremoval.deleteTask(options)</a>	void	Server scripts	Deletes a personal information removal task.
	<a href="#">piremoval.getTaskStatus(options)</a>	<a href="#">piremoval.PiRemovalTaskStatus</a>	Server scripts	Retrieves the status of a personal information removal task.
	<a href="#">piremoval.loadTask(options)</a>	<a href="#">piremoval.PiRemovalTask</a>	Server scripts	Loads a personal information removal task.

## PiRemovalTask Object Members

The following members are called on the [piremoval.PiRemovalTask](#) object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">PiRemovalTask.deleteTask()</a>	void	Server scripts	Deletes the personal information removal task.
	<a href="#">PiRemovalTask.run()</a>	void	Server scripts	Runs the personal information removal task.
	<a href="#">PiRemovalTask.save()</a>	void	Server scripts	Saves the personal information removal task.
Property	<a href="#">PiRemovalTask.fieldIds</a>	string[] (read-only)	Server scripts	Represents the field IDs that are processed by the PI removal task.
	<a href="#">PiRemovalTask.historyOnly</a>	boolean	Server scripts	Indicates whether the PI removal task removes system note information only, not field values or workflow history.
	<a href="#">PiRemovalTask.historyReplacement</a>	string (read-only)	Server scripts	Represents the text used in system notes to replace the original values.
	<a href="#">PiRemovalTask.id</a>	number (read-only)	Server scripts	Represents the ID of the personal information removal task.
	<a href="#">PiRemovalTask.recordIds</a>	number[] (read-only)	Server scripts	Represents the record IDs that are processed by the PI removal task.
	<a href="#">PiRemovalTask.recordType</a>	string (read-only)	Server scripts	Describes the record type updated by the PI removal task.
	<a href="#">PiRemovalTask.status</a>	<a href="#">piremoval.PiRemovalTaskStatus</a>	Server scripts	Describes the status of the submitted personal information removal task.
	<a href="#">PiRemovalTask.workflowIds</a>	number[] (read-only)	Server scripts	Represents the workflow IDs whose history is processed by the PI removal task.

## PiRemovalTaskLogItem Object Members

The following members are called on the [piremoval.PiRemovalTaskLogItem](#) object.

Member Type	Name	Return Type/Value Type	Support Script Type	Description
Property	<a href="#">PiRemovalTaskLogItem.exception</a>	string (read-only)	Server scripts	Describes the exception for the log item, including and what caused it.
	<a href="#">PiRemovalTaskLogItem.message</a>	string (read-only)	Server scripts	Describes the message for the log item and an explanation for any errors.

Member Type	Name	Return Type/Value Type	Support Script Type	Description
	PiRemovalTaskLogItem.status	string (read-only)	Server scripts	Describes the status of the log item. This property takes its values from <a href="#">task.TaskStatus</a> .
	PiRemovalTaskLogItem.type	string (read-only)	Server scripts	Describes the type of personal information that was removed, one of <a href="#">FieldValue</a> , <a href="#">SystemNote</a> , or <a href="#">Workflow</a> .

## PiRemovalTaskStatus Object Members

The following members are called on the [piremoval.PiRemovalTaskStatus](#) object.

Member Type	Name	Return Type/Value Type	Support Script Type	Description
Property	PiRemovalTaskStatus.logList	list	Server scripts	Represents a list of logs for the PI removal task job.
	PiRemovalTaskStatus.status	string	Server scripts	Describes the status of the submitted personal information removal task.

## N/piremoval Module Script Samples

The following script samples demonstrate how to use the features of the N/piremoval module.

### Sample 1: Remove customer phone number

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to remove the phone numbers and comments in specific customer records from the record fields (field values), system notes, and workflow history. The removed values are replaced with `removed_value`.

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

require(['N/piremoval'], function(piremoval) {
    function removePersonalInformation() {
        var piRemovalTask = piremoval.createTask({
            recordType: 'customer',
            recordIds: [11, 19],
            fieldIds: ['comments', 'phone'],
            workflowIds: [1],
            historyOnly: false,
        });
    }
});
```

```

        historyReplacement: 'removed_value'
    });

    piRemovalTask.save();
    var taskId = piRemovalTask.id;

    var piRemovalTaskInProgress = piremoval.loadTask(taskId);
    piRemovalTaskInProgress.run();

    var status = piremoval.getTaskStatus(taskId);
};

removePersonalInformation();
});

```

## piremoval.PiRemovalTask



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a task to remove personal information (PI). This object includes lists of the record, field, and workflow IDs to remove personal information from, as well as history replacement information.  Use <a href="#">piremoval.createTask(options)</a> to create a <code>piremoval.PiRemovalTask</code> object, or use <a href="#">piremoval.loadTask(options)</a> to load a PI removal task as a <code>piremoval.PiRemovalTask</code> object. To save the object, use <a href="#">PiRemovalTask.save()</a> . To execute the PI removal, use <a href="#">PiRemovalTask.run()</a> .
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/piremoval Module</a>
<b>Methods and Properties</b>	<a href="#">PiRemovalTask Object Members</a>
<b>Since</b>	2019.2

## Syntax



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

```

// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [11, 19],
    fieldIds: ['comments', 'phone'],
    workflowIds: [1],
    historyOnly: false,
    historyReplacement: 'removed_value'
});

```

```

myPiRemovalTask.save();
var myTaskId = myPiRemovalTask.id;

myPiRemovalTask.run();
...
// Add additional code

```

## PiRemovalTask.deleteTask()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Deletes the PI Removal task.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	20 units
<b>Module</b>	<a href="#">N/piremoval Module</a>
<b>Parent Object</b>	<a href="#">piremoval.PiRemovalTask</a>
<b>Sibling Object Members</b>	<a href="#">PiRemovalTask Object Members</a>
<b>Since</b>	2019.2

## Errors

Error Code	Error Message	Thrown If
UNEXPECTED_ERROR	Cannot delete PiRemoval job that was not saved.	The PI removal job is not saved.

## Syntax



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

```

// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [11, 19],
    fieldIds: ['comments', 'phone'],
    workflowIds: [1],
    historyOnly: false,
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();

```

```
var myTaskId = myPiRemovalTask.id;

myPiRemovalTask.deleteTask();
...
// Add additional code
```

## PiRemovalTask.run()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Runs the PI removal task.  All validation for the task (for example, ensuring that the specified record IDs are valid) occurs when the task is saved using <a href="#">PiRemovalTask.save()</a> , not when the task is run using <a href="#">PiRemovalTask.run()</a> .
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	20 units
<b>Module</b>	<a href="#">N/piremoval Module</a>
<b>Parent Object</b>	<a href="#">piremoval.PiRemovalTask</a>
<b>Sibling Object Members</b>	<a href="#">PiRemovalTask Object Members</a>
<b>Since</b>	2019.2

## Errors

Error Code	Error Message	Thrown If
UNEXPECTED_ERROR	Cannot run unsaved PiRemoval job.	The PI removal job is not saved.

## Syntax

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

```
// Add additional code
...

var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [11, 19],
    fieldIds: ['comments', 'phone'],
    workflowIds: [1],
    historyOnly: false,
    historyReplacement: 'removed_value'
});
```

```

myPiRemovalTask.save();
var myTaskId = myPiRemovalTask.id;

myPiRemovalTask.run();
...
// Add additional code

```

## PiRemovalTask.save()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Saves the PI removal task.  All validation for the task (for example, ensuring that the specified record IDs are valid) occurs when the task is saved using this method.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	20 units
<b>Module</b>	<a href="#">N/piremoval Module</a>
<b>Parent Object</b>	<a href="#">piremoval.PiRemovalTask</a>
<b>Sibling Object Members</b>	<a href="#">PiRemovalTask Object Members</a>
<b>Since</b>	2019.2

## Errors

Error Code	Error Message	Thrown If
_1_CANNOT_BE_EMPTY	Record Type cannot be empty.	The record type is not set.
_1_JOB_WAS_NOT_FOUND	Record Type 'type' was not found.	Record type does not exist.
_1_JOB_WAS_NOT_FOUND	Record ID 'ID' was not found.	One of the record IDs does not exist.
_1_JOB_WAS_NOT_FOUND	Field ID 'ID' was not found.	One of the field IDs does not exist.
_1_JOB_WAS_NOT_FOUND	Workflow ID 'ID' was not found.	One of the workflow IDs does not exist.

## Syntax



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

```

// Add additional code
...
var myPiRemovalTask = piremoval.createTask({

```

```

    recordType: 'customer',
    recordIds: [11, 19],
    fieldIds: ['comments', 'phone'],
    workflowIds: [1],
    historyOnly: false,
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var mytaskId = myPiRemovalTask.id;

myPiRemovalTask.run();
...
// Add additional code

```

## PiRemovalTask.fieldIds



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	IDs of the fields whose PI is removed.  If no field IDs are entered, no information changes are performed. If the field IDs are null or invalid, the following exception occurs: <ul style="list-style-type: none"><li>■ Wrong parameter type: options.fieldIds is expected as array.</li></ul>
<b>Type</b>	string[] (read-only)
<b>Module</b>	<a href="#">N/piremoval Module</a>
<b>Parent Object</b>	<a href="#">piremoval.PiRemovalTask</a>
<b>Sibling Object Members</b>	<a href="#">PiRemovalTask Object Members</a>
<b>Since</b>	2019.2

## Syntax



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

```

// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],
    fieldIds: ['email'],
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var theFieldIds = myPiRemovalTask.fieldIds;
...
// Add additional code

```

## PiRemovalTask.historyOnly

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether the PI removal task removes system note information only, not field values or workflow history. If <code>true</code> , the task removes information from system notes only. If <code>false</code> , the task removes information from system notes, workflow history, and field values. The default value is <code>false</code> .
<b>Type</b>	boolean (read-only)
<b>Module</b>	<a href="#">N/piremoval Module</a>
<b>Parent Object</b>	<a href="#">PiRemovalTask</a>
<b>Sibling Object Members</b>	<a href="#">PiRemovalTask Object Members</a>
<b>Since</b>	2019.2

## Syntax

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

```
// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],
    fieldIds: ['email'],
    historyOnly: true,
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var theHistoryOnly = myPiRemovalTask.historyOnly;
...
// Add additional code
```

## PiRemovalTask.historyReplacement

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The text used in system notes to replace the original value.
<b>Type</b>	string (read-only)
<b>Module</b>	<a href="#">N/piremoval Module</a>
<b>Parent Object</b>	<a href="#">PiRemovalTask</a>
<b>Sibling Object Members</b>	<a href="#">PiRemovalTask Object Members</a>
<b>Since</b>	2019.2

## Syntax



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

```
// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],
    fieldIds: ['email'],
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var theHistoryReplacement = myPiRemovalTask.historyReplacement;
...
// Add additional code
```

## PiRemovalTask.id



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	ID that uniquely identifies the PI removal task. This ID is assigned to the PI removal task when <a href="#">PiRemovalTask.save()</a> is called to save the task. You cannot specify your own task ID.
<b>Type</b>	number (read-only)
<b>Module</b>	N/piremoval Module
<b>Parent Object</b>	<a href="#">PiRemovalTask</a>
<b>Sibling Object Members</b>	<a href="#">PiRemovalTask Object Members</a>
<b>Since</b>	2019.2

## Syntax



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

```
// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],
    fieldIds: ['email'],
    historyReplacement: 'removed_value'
};

myPiRemovalTask.save();
var theId = myPiRemovalTask.id;
```

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

## PiRemovalTask.recordIds



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	ID of records whose PI is removed.  If no record IDs are entered, no information changes are performed. If the record IDs are null or invalid, the following exception occurs: <ul style="list-style-type: none"><li>■ Wrong parameter type: options.recordIds is expected as array.</li></ul>
<b>Type</b>	number[] (read-only)
<b>Module</b>	N/piremoval Module
<b>Parent Object</b>	piremoval.PiRemovalTask
<b>Sibling Object Members</b>	<a href="#">PiRemovalTask Object Members</a>
<b>Since</b>	2019.2

## Syntax



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

```
// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95, 107],
    fieldIds: ['email'],
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var theRecordIds = myPiRemovalTask.recordIds;
...
// Add additional code
```

## PiRemovalTask.recordType



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Type of record whose PI is removed. All records referenced in the piremoval.PiRemovalTask object must be the same type.
<b>Type</b>	string (read-only)
<b>Module</b>	N/piremoval Module

<b>Parent Object</b>	piremoval.PiRemovalTask
<b>Sibling Object Members</b>	PiRemovalTask Object Members
<b>Since</b>	2019.2

## Syntax

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

```
// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],
    fieldIds: ['email'],
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var theRecordType = myPiRemovalTask.recordType;
...
// Add additional code
```

## PiRemovalTask.status

**ℹ Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Status of the PI removal task.
<b>Type</b>	piremoval.PiRemovalTaskStatus
<b>Module</b>	N/piremoval Module
<b>Parent Object</b>	piremoval.PiRemovalTask
<b>Sibling Object Members</b>	PiRemovalTask Object Members
<b>Since</b>	2019.2

## Syntax

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

```
// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],
    fieldIds: ['email'],
    historyReplacement: 'removed_value'
});
```

```
myPiRemovalTask.save();
var theStatus = myPiRemovalTask.status;
...
// Add additional code
```

## PiRemovalTask.workflowIds

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	IDs of workflows where PI is removed from the workflow history. If no workflow IDs are entered, no information changes are performed. If the workflow IDs are null or invalid, the following exception occurs: <ul style="list-style-type: none"><li>■ Wrong parameter type: options.workflowIds is expected as array.</li></ul>
<b>Type</b>	number[] (read-only)
<b>Module</b>	N/piremoval Module
<b>Parent Object</b>	<a href="#">piremoval.PiRemovalTask</a>
<b>Sibling Object Members</b>	<a href="#">PiRemovalTask Object Members</a>
<b>Since</b>	2019.2

## Syntax

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

```
// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95, 107],
    fieldIds: ['email', 'phone'],
    workflowIds: [1, 7],
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var theWorkflowIds = myPiRemovalTask.workflowIds;
...
// Add additional code
```

## piremoval.PiRemovalTaskStatus

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the status of a personal information removal task returned by <a href="#">piremoval.getTaskStatus(options)</a> .
---------------------------	---

	<p>Possible status values include:</p> <ul style="list-style-type: none"> <li>■ PENDING</li> <li>■ PROCESSING</li> <li>■ COMPLETE</li> <li>■ FAILED</li> </ul> <p>For more information, see <a href="#">task.TaskStatus</a>.</p>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/piremoval Module</a>
<b>Methods and Properties</b>	<a href="#">PiRemovalTaskStatus Object Members</a>
<b>Since</b>	2019.2

## Syntax

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

```
// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],
    fieldIds: ['email'],
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var myId = myPiRemovalTask.id;

var myFirstStatus = piremoval.getTaskStatus({
    id: myId
});

myPiRemovalTask.run();

var mySecondStatus = piremoval.getTaskStatus({
    id: myId
});
...
// Add additional code
```

## PiRemovalTaskStatus.logList

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Gets list of logs for the PiRemovalTask job.
<b>Type</b>	<a href="#">piremoval.PiRemovalTaskLogItem[]</a>

<b>Module</b>	N/piremoval Module
<b>Parent Object</b>	piremoval.PiRemovalTaskStatus
<b>Sibling Object Members</b>	PiRemovalTaskStatus Object Members
<b>Since</b>	2019.2

## Syntax

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

```
// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],
    fieldIds: ['email'],
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var myId = myPiRemovalTask.id;

var myStatus = piremoval.getTaskStatus({
    id: myId
});

var theLogList = myStatus.logList;
for (var i = 0; i < theLogList.length; i++) {
    log.debug({
        title: 'logList value',
        details: theLogList[i]
    });
}
...
// Add additional code
```

## PiRemovalTaskStatus.status

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>Encapsulates the status of a personal information removal task returned by <a href="#">piremoval.getTaskStatus(options)</a>.</p> <p>Possible status values include:</p> <ul style="list-style-type: none"> <li>■ PENDING</li> <li>■ PROCESSING</li> <li>■ COMPLETE</li> <li>■ FAILED</li> </ul> <p>For more information, see <a href="#">task.TaskStatus</a>.</p>
-----------------------------	--

Type	string (read-only)
Module	N/piremoval Module
Parent Object	piremoval.PiRemovalTaskStatus
Sibling Object Members	<a href="#">PiRemovalTaskStatus Object Members</a>
Since	2019.2

## Syntax

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

```
// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],
    fieldIds: ['email'],
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var myId = myPiRemovalTask.id;

var myTaskStatus = piremoval.getTaskStatus({
    id: myId
});
var theStatus = myTaskStatus.status;
...
// Add additional code
```

## piremoval.PiRemovalTaskLogItem

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	This object represents log items that are associated with a <a href="#">piremoval.PiRemovalTaskStatus</a> object. The logs are generated separately when a task is created, started, and completed. A <a href="#">piremoval.PiRemovalTaskLogItem</a> object represents a single log entry that you get from the <a href="#">piremoval.PiRemovalTaskStatus</a> object using the <a href="#">PiRemovalTaskStatus.logList</a> property. The log items are sorted by date.  The structure of this object is described in <a href="#">PiRemovalTaskLogItem Object Members</a> .
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/piremoval Module</a>
<b>Methods and Properties</b>	<a href="#">PiRemovalTaskLogItem Object Members</a>
<b>Since</b>	2019.2

## Syntax

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

```
// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],
    fieldIds: ['email'],
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var myId = myPiRemovalTask.id;

var myStatus = piremoval.getTaskStatus({
    id: myId
});

var theLogList = myStatus.logList;
for (var i = 0; i < theLogList.length; i++) {
    log.debug({
        title: 'logList value',
        details: theLogList[i]
    });
}
...
// Add additional code
```

## PiRemovalTaskLogItem.exception

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Exception message for the log item, typically an unexpected error from NetSuite.
<b>Type</b>	string (read-only)
<b>Module</b>	<a href="#">N/piremoval Module</a>
<b>Parent Object</b>	<a href="#">piremoval.PiRemovalTaskLogItem</a>
<b>Sibling Object Members</b>	<a href="#">PiRemovalTaskLogItem Object Members</a>
<b>Since</b>	2019.2

## Syntax

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

```
// Add additional code
```

```

...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],
    fieldIds: ['email'],
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var myId = myPiRemovalTask.id;

var myStatus = piremoval.getTaskStatus({
    id: myId
});

var theLogList = myStatus.logList;
for (var i = 0; i < theLogList.length; i++) {
    var theLogListException = theLogList[i].exception;

    // Do something with the log list exception here
}
...
// Add additional code

```

## PiRemovalTaskLogItem.message



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Log item text message. The message specifies if the record type is not set, or if one of record, field, or workflow IDs do not exist.
<b>Type</b>	string (read-only)
<b>Module</b>	N/piremoval Module
<b>Parent Object</b>	<a href="#">piremoval.PiRemovalTaskLogItem</a>
<b>Sibling Object Members</b>	<a href="#">PiRemovalTaskLogItem Object Members</a>
<b>Since</b>	2019.2

## Syntax



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

```

// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],
    fieldIds: ['email'],

```

```

        historyReplacement: 'removed_value'
    });

myPiRemovalTask.save();
var myId = myPiRemovalTask.id;

var myStatus = piremoval.getTaskStatus({
    id: myId
});

var theLogList = myStatus.logList;
for (var i = 0; i < theLogList.length; i++) {
    var theLogListMessage = theLogList[i].message;

    // Do something with the log list message here
}

...
// Add additional code

```

## PiRemovalTaskLogItem.status

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Encapsulates the status of a log item.  Possible status values include: <ul style="list-style-type: none"><li>■ PENDING</li><li>■ PROCESSING</li><li>■ COMPLETE</li><li>■ FAILED</li></ul> For more information, see <a href="#">task.TaskStatus</a> .
<b>Type</b>	string (read-only)
<b>Module</b>	N/piremoval Module
<b>Parent Object</b>	piremoval.PiRemovalTaskLogItem
<b>Sibling Object Members</b>	<a href="#">PiRemovalTaskLogItem Object Members</a>
<b>Since</b>	2019.2

## Syntax

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

```

// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],

```

```

    fieldIds: ['email'],
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var myId = myPiRemovalTask.id;

var myStatus = piremoval.getTaskStatus({
    id: myId
});

var theLogList = myStatus.logList;
for (var i = 0; i < theLogList.length; i++) {
    var theLogListStatus = theLogList[i].status;

    // Do something with the log list status here
}
...
// Add additional code

```

## PiRemovalTaskLogItem.type

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates the change described by this log item. Possible values include: <ul style="list-style-type: none"><li>■ FieldValue - field value</li><li>■ SystemNote - system note</li><li>■ Workflow - workflow history</li></ul>
<b>Type</b>	string (read-only)
<b>Module</b>	N/piremoval Module
<b>Parent Object</b>	piremoval.PiRemovalTaskLogItem
<b>Sibling Object Members</b>	<a href="#">PiRemovalTaskLogItem Object Members</a>
<b>Since</b>	2019.2

## Syntax

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

```

// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],
    fieldIds: ['email'],
    historyReplacement: 'removed_value'
});

```

```

myPiRemovalTask.save();
var myId = myPiRemovalTask.id;

var myStatus = piremoval.getTaskStatus({
    id: myId
});

var theLogList = myStatus.logList;
for (var i = 0; i < theLogList.length; i++) {
    var theLogListType = theLogList[i].type;

    // Do something with the log list type here
}

...
// Add additional code

```

## piremoval.createTask(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a new personal information removal task.
<b>Returns</b>	<a href="#">piremoval.PiRemovalTask</a>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/piremoval Module</a>
<b>Sibling Object Members</b>	<a href="#">N/piremoval Module Members</a>
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.fieldIds</code>	number[]	optional	Represents IDs of fields whose personal information is removed.
<code>options.historyOnly</code>	boolean	optional	Indicates whether the PI removal task removes system note information only, not field values or workflow history. If true, the task removes information from system notes only. If false, the task removes information from system notes, workflow history, and field values. The default value is false.
<code>options.historyReplacement</code>	string	optional	Represents the text used in system notes to replace the original values.

Parameter	Type	Required / Optional	Description
options.recordIds	number[]	optional	Represents IDs of records whose personal information is removed.
options.recordType	string	optional	Describes the record type that is updated by the PI removal task.
options.workflowIds	number[]	optional	Represents the workflow IDs whose history is processed by the PI removal task.

## Syntax

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

```
// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95, 107],
    fieldIds: ['email', 'phone'],
    workflowIds: [3, 7],
    historyOnly: true,
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var myTaskId = myPiRemovalTask.id;

myPiRemovalTask.run();
...
// Add additional code
```

## piremoval.deleteTask(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Deletes a personal information removal task.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	20 units
<b>Module</b>	<a href="#">N/piremoval Module</a>
<b>Sibling Object Members</b>	<a href="#">N/piremoval Module Members</a>
<b>Since</b>	2019.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description
options.id	number	required	Unique identifier of the personal information removal task.

## Errors

Error Code	Error Message	Thrown If
UNEXPECTED_ERROR	Cannot delete PIRemoval job that was not saved	Job is not saved.

## Syntax

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

```
// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95, 107],
    fieldIds: ['email', 'phone'],
    workflowIds: [3, 7],
    historyOnly: true,
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var myTaskId = myPiRemovalTask.id;

piremoval.deleteTask({
    id: myTaskId
});
...
// Add additional code
```

## piremoval.getTaskStatus(options)

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Retrieves the status of a personal information removal task.
<b>Returns</b>	<a href="#">piremoval.PiRemovalTaskStatus</a>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Governance</b>	None
<b>Module</b>	N/piremoval Module
<b>Sibling Object Members</b>	N/piremoval Module Members
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.id</code>	number	required	Unique identifier of the personal information removal task.

## Syntax

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

```
// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],
    fieldIds: ['email'],
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var myId = myPiRemovalTask.id;

var myFirstStatus = piremoval.getTaskStatus({
    id: myId
});

myPiRemovalTask.run();

var mySecondStatus = piremoval.getTaskStatus({
    id: myId
});
...
// Add additional code
```

## piremoval.loadTask(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Retrieves an existing personal information removal task.
---------------------------	--

<b>Returns</b>	piremoval.PiRemovalTask
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/piremoval Module
<b>Sibling Object Members</b>	<a href="#">N/piremoval Module Members</a>
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.id	number	required	Unique identifier of the personal information removal task.

## Errors

Error Code	Error Message	Thrown If
_1_WAS_NOT_FOUND	PIRemoval job was not found.	Job with the ID provided was not found.

## Syntax

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

```
// Add additional code
...
var myPiRemovalTask = piremoval.createTask({
    recordType: 'customer',
    recordIds: [95],
    fieldIds: ['email'],
    historyReplacement: 'removed_value'
});

myPiRemovalTask.save();
var myId = myPiRemovalTask.id;

var myLoadedPiRemovalTask = piremoval.loadTask({
    id: myId
});

myLoadedPiRemovalTask.run();
...
// Add additional code
```

# N/plugin Module



**Note:** The content in this help topic pertains to SuiteScript 2.0.

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 on ad-hoc and deployed debugging, see the help topic [SuiteScript Debugger](#).

- [N/plugin Module Members](#)
- [N/plugin Module Script Samples](#)

## N/plugin Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">plugin.findImplementations(options)</a>	string[]	Server-side scripts	Returns the script IDs of custom plug-in type implementations.
Method	<a href="#">plugin.loadImplementation(options)</a>	Object	Server-side scripts	Instantiates an implementation of the custom plug-in type.

## N/plugin Module Script Samples

The following script samples demonstrate how to use the features of the N/plugin module.

### Sample 1: Find plugin implementations



**Note:** This script sample uses the `define` function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the `require` function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample shows an implementation of the plugin interface. To test this sample, you need a custom plugin type with a script ID of `customscript_magic_plugin` and an interface with a single method, `int doTheMagic(int, int)`.

```
/*
 * @NApiVersion 2.0
 * @NScriptType plugintypeimpl
 */

define(function() {
    return {
        doTheMagic: function (operand1, operand2) {
            return operand1 + operand2;
        }
    }
})
```

```

    }
});
```

The following Suitelet iterates through all implementations of the custom plugin type `customscript_magic_plugin`. For the plugin to be recognized, the Suitelet script record must specify the plugin type under Custom Plug-in Types.

```

/**
 * @NApiVersion 2.x
 * @NScriptType Suitelet
 */
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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the script IDs of custom plug-in type implementations.  Returns an empty list when there is no custom plug-in type with the script ID available for the executing script.
<b>Returns</b>	A string[] containing a list of custom plug-in implementation script IDs.
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	
<b>Module</b>	<a href="#">N/config Module</a>
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The script ID of the custom plug-in type.	2016.1
options.includeDefault	boolean true   false	optional	The default value is true, indicating that the default implementation should be included in the list.	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/plugin Module Script Sample.

```
//Add additional code
...
var impls = plugin.findImplementations({
    type: 'customscript_sample_plugin'
});
...
//Add additional code
```

## plugin.loadImplementation(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	An Object implementing the custom plug-in type.
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	
<b>Module</b>	<a href="#">N/config Module</a>
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The script ID of the custom plug-in type.	2016.1

Parameter	Type	Required / Optional	Description	Since
options.implementation	string	optional	The script ID of the custom plug-in implementation.	2016.1

Error Code	Thrown If
UNABLE_TO_FIND_IMPLEMENTATION_1_FOR_PLUGIN_2	Either there is no such implementation of the provided plug-in type, or the plug-in type does not exist.

## Syntax

**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 Sample.

```
//Add additional code
...
var pl = plugin.loadImplementation({
    type: 'customscript_sample_plugin'
});
...
//Add additional code
```

## N/portlet Module

**Note:** The content in this help topic pertains to SuiteScript 2.0.

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

Member Type	Name	Return Type	Supported Script Types	Description
Method	portlet.resize	void	Client scripts	Resizes a form portlet immediately.
	portlet.refresh	void	Client scripts	Refreshes a form portlet immediately.

## N/portlet Module Script Sample

**Note:** This script sample uses the **define** function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the **require** function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

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.

This 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](#).

```
/*
 * @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({
                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
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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Resizes a form portlet type immediately.
<b>Returns</b>	Void

<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/portlet Module</a>
<b>Since</b>	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](#)

```
//Add additional code
...
portlet.resize();
...
//Add additional code
```

## portlet.refresh

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Refreshes a form portlet type immediately.
<b>Returns</b>	Void
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/portlet Module</a>
<b>Since</b>	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](#)

```
...
portlet.refresh();
...
```

## N/query Module

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

Load the N/query module to create and run queries using the SuiteAnalytics Workbook query engine. For more information about SuiteAnalytics Workbook, see the help topic [SuiteAnalytics Workbook Overview](#).

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.
- Convert query objects to SuiteQL queries and run arbitrary SuiteQL queries.

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](#)
- [SuiteQL 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. 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 Record Types](#).

## In This Help Topic

- [N/query Module Members](#)
- [Column Object Members](#)
- [Component Object Members](#)
- [Condition Object Members](#)
- [Page Object Members](#)
- [PagedData Object Members](#)
- [PageRange Object Members](#)
- [Query Object Members](#)
- [RelativeDate Object Members](#)
- [Result Object Members](#)
- [ResultSet Object Members](#)
- [Sort Object Members](#)
- [SuiteQL Object Members](#)
- [N/query Module Script Samples](#)

## N/query Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	query.Column	Object	Client and server scripts	<p>The field types (query result columns) that are displayed from the query results.</p> <p>Use <a href="#">Query.createColumn(options)</a> or <a href="#">Component.createColumn(options)</a> to create this object.</p>
	query.Component	Object	Client and server scripts	<p>One component of the query definition. The query definition always contains at least one component that encapsulates the initial query type. Queries with joins contain multiple components that encapsulate the join relationships.</p> <p>The initial component (<a href="#">Query.root</a>) is automatically created with the query definition (<a href="#">query.Query</a>). Use <a href="#">Query.autoJoin(options)</a> or <a href="#">Component.autoJoin(options)</a> to create subsequent components.</p>
	query.Condition	Object	Client and server scripts	<p>A condition. A condition narrows the query results.</p> <p>Use <a href="#">Query.createCondition(options)</a> or <a href="#">Component.createCondition(options)</a> to create this object.</p>
	query.Page	Object	Client and server scripts	One page of the paged query results.
	query.PagedData	Object	Client and server scripts	A set of paged query results. This object also contains information about the set of paged results it encapsulates.
	query.PageRange	Object	Client and server scripts	A range of pages from the paged query results.
	query.RelativeDate	Object	Client and server scripts	A relative date to use in query conditions.
	query.Result	Object	Client and server scripts	A single row of the query result set.
	query.ResultSet	Object	Client and server scripts	The set of results returned by the query.
	query.Query	Object	Client and server scripts	<p>The query definition. Use <a href="#">query.create(options)</a> or <a href="#">query.load(options)</a> to create this object.</p> <p><b>The creation of this object is the first step in creating a query with the N/query Module.</b></p>
Object	query.Sort	Object	Client and server scripts	<p>A sort that is placed on a particular query result column.</p> <p>Use <a href="#">Query.createSort(options)</a> or <a href="#">Component.createSort(options)</a> to create this object.</p>
	query.SuiteQL	Object	Client and server scripts	A SuiteQL query.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
				Use <a href="#">Query.toSuiteQL()</a> to create this object.
Method	<a href="#">query.create(options)</a>	<a href="#">query.Query</a>	Client and server scripts	Creates the query definition. <b>The execution of this method is the first step in creating a query with the N/query Module.</b>
	<a href="#">query.createRelativeDate(options)</a>	<a href="#">query.RelativeDate</a>	Client and server scripts	Creates a <a href="#">query.RelativeDate</a> object that represents a date relative to the current date.
	<a href="#">query.delete(options)</a>	void	Client and server scripts	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.
	<a href="#">query.load(options)</a>	<a href="#">query.Query</a>	Client and server scripts	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 query types, and executed in the same way as queries created using <a href="#">query.create(options)</a> .
	<a href="#">query.load.promise(options)</a>	<a href="#">query.Query</a>	Client scripts	Asynchronously loads an existing query that was created using the SuiteAnalytics Workbook UI.
	<a href="#">query.runSuiteQL(options)</a>	<a href="#">query.ResultSet</a>	Client and server scripts	Runs an arbitrary SuiteQL query.
	<a href="#">query.runSuiteQLPaged(options)</a>	<a href="#">query.PagedData</a>	Client and server scripts	Runs an arbitrary SuiteQL query as a paged query.
Enum	<a href="#">query.Aggregate</a>	enum	Client and server scripts	Holds the string values for aggregate functions supported with the <a href="#">N/query Module</a> .  This enum is used to pass the aggregate function argument to <a href="#">Component.createColumn(options)</a> , <a href="#">Component.createCondition(options)</a> , <a href="#">Query.createColumn(options)</a> , and <a href="#">Query.createCondition(options)</a> .
	<a href="#">query.DateId</a>	enum	Client and server scripts	Holds the string values for supported date codes in relative dates.  This enum is used to pass the date ID argument to <a href="#">query.createRelativeDate(options)</a> .
	<a href="#">query.FieldContext</a>	enum	Client and server scripts	Holds the string values for the field context to use when creating a column.  This enum is used to pass the context argument to <a href="#">Query.createColumn(options)</a> and <a href="#">Component.createColumn(options)</a> .
	<a href="#">query.Operator</a>	enum	Client and server scripts	Holds the string values for operators supported with the <a href="#">N/query Module</a> .  This enum is used to pass the operator argument to <a href="#">Query.createCondition(options)</a> and <a href="#">Component.createCondition(options)</a> .

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	query.RelativeDateRange	enum	Client and server scripts	<p>Holds <a href="#">query.RelativeDate</a> object values for supported date ranges in relative dates.</p> <p>This enum is used to pass the values argument to <a href="#">Query.createCondition(options)</a> and <a href="#">Component.createCondition(options)</a>.</p>
	query.ReturnType	enum	Client and server scripts	<p>Holds the string values for the formula return types supported with the <a href="#">N/query Module</a>.</p> <p>This enum is used to pass the formula return type argument to <a href="#">Query.createColumn(options)</a>, <a href="#">Component.createColumn(options)</a>, <a href="#">Query.createCondition(options)</a>, and <a href="#">Component.createCondition(options)</a>.</p>
	query.SortLocale	enum	Client and server scripts	<p>Holds the string values for sort locales supported with the <a href="#">N/query Module</a>.</p> <p>This enum is used to pass the sort locale argument to <a href="#">Query.createSort(options)</a> and <a href="#">Component.createSort(options)</a>.</p>
	query.Type	enum	Client and server scripts	<p>Holds the string values for supported query types used in the query definition.</p> <p>This enum is used to pass the initial query type argument to <a href="#">query.create(options)</a>.</p>

## Column Object Members

The following members are called on the [query.Column](#) object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Property	Column.aggregate	string (read-only)	Client and server scripts	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.
	Column.alias	string (read-only)	Client and server scripts	An alias for this column. An alias is an alternate name for a column, and the alias is used in mapped results.
	Column.component	<a href="#">query.Component</a> (read-only)	Client and server scripts	A reference to the <a href="#">query.Component</a> object to which this query result column belongs.
	Column.context	Object (read-only)	Client and server scripts	The field context for values in the query result column. The field context determines how field values are displayed in the column.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
	Column.fieldId	string (read-only)	Client and server scripts	The name of the query result column. This property and the <a href="#">Column.formula</a> property cannot be set at the same time.
	Column.formula	string (read-only)	Client and server scripts	The formula used to create the query result column. This property and the <a href="#">Column.fieldId</a> property cannot be set at the same time.
	Column.groupBy	boolean (read-only)	Client and server scripts	Whether the query results are grouped by this query result column.
	Column.type	string (read-only)	Client and server scripts	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	Name	Return Type/Value Type	Supported Script Types	Description
Method	Component.autoJoin(options)	query.Component	Client and server scripts	<p>Creates a join relationship.</p> <p>After you create the initial query definition, use <a href="#">Query.autoJoin(options)</a> to create your first join. Then use this method to create each subsequent join.</p> <p>This method selects the correct join type automatically based on the record types that are being joined.</p>
	Component.createColumn(options)	query.Column	Client and server scripts	<p>Creates a query result column based on the component.</p> <p>Use this method to create columns based on the join relationships created with <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autoJoin(options)</a>.</p>
	Component.createCondition(options)	query.Condition	Client and server scripts	<p>Creates a condition (filter column) based on the component.</p> <p>Use this method to create conditions based on the join relationships created with <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autoJoin(options)</a>.</p>
	Component.createSort(options)	query.Sort	Client and server scripts	<p>Creates a sort based on the component.</p> <p>Use this method to create sorts based on the join relationships created with <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autoJoin(options)</a>.</p>
	Component.join(options)	query.Component	Client and server scripts	Creates a join relationship. This method is an alias to <a href="#">Component.autoJoin(options)</a> .

Member Type	Name	Return Type/ Value Type	Supported Script Types	Description
				After you create the initial query definition, use <a href="#">Query.autoJoin(options)</a> to create your first join. Then use this method, or <a href="#">Component.autoJoin(options)</a> , to create each subsequent join.
	<a href="#">Component.joinFrom(options)</a>	<a href="#">query.Component</a>	Client and server scripts	<p>Creates an explicit directional join relationship from another component to this component (an inverse join). This method sets the <a href="#">Component.source</a> property on the returned <a href="#">query.Component</a> object.</p> <p>After you create the initial query definition, use this method to create explicit directional joins from other components to this component.</p>
	<a href="#">Component.joinTo(options)</a>	<a href="#">query.Component</a>	Client and server scripts	<p>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 query types. This method sets the <a href="#">Component.target</a> property on the returned <a href="#">query.Component</a> object.</p> <p>After you create the initial query definition, use this method to create explicit directional joins to other components from this component.</p>
Property	<a href="#">Component.child</a>	Object (read-only)	Client and server scripts	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 <a href="#">query.Component</a> object.
	<a href="#">Component.parent</a>	string (read-only)	Client and server scripts	The parent <a href="#">query.Component</a> object of the component.
	<a href="#">Component.source</a>	string (read-only)	Client and server scripts	The source query type of the component. The value of this property is set when <a href="#">Component.joinFrom(options)</a> is called to perform an explicit directional join from another component.
	<a href="#">Component.target</a>	string (read-only)	Client and server scripts	The target query type of the component. The value of this property is set when <a href="#">Component.joinTo(options)</a> is called to perform an explicit directional join to another component.
	<a href="#">Component.type</a>	string (read-only)	Client and server scripts	The query type of the component.

## Condition Object Members

The following members are called on the [query.Condition](#) object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Property	Condition.aggregate	string (read-only)	Client and server scripts	An aggregate function that is performed on the condition. An aggregate function performs a calculation on the condition values and returns a single value.
	Condition.children	query.Condition[] (read-only)	Client and server scripts	An array of child conditions used to create the parent condition.
	Condition.component	query.Component (read-only)	Client and server scripts	A reference to the query.Component object to which this condition belongs.
	Condition.fieldID	string (read-only)	Client and server scripts	The name of the field that is used in the condition.
	Condition.formula	string (read-only)	Client and server scripts	The formula used to create the condition.
	Condition.operator	string (read-only)	Client and server scripts	The name of the operator used to create the condition.
	Condition.type	string (read-only)	Client and server scripts	The return type of the formula used to create the condition.
	Condition.values	string   number   boolean   Array<string   number   boolean> (read-only)	Client and server scripts	An array of values used by an operator to create the condition.

## Page Object Members

The following members are called on the [query.Page](#) object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Property	Page.data	query.ResultSet (read-only)	Client and server scripts	The query results contained in this page.
	Page.isFirst	boolean (read-only)	Client and server scripts	Whether this page is the first of the paged query results.
	Page.isLast	boolean (read-only)	Client and server scripts	Whether this page is the last of the paged query results.
	Page.pagedData	query.PagedData (read-only)	Client and server scripts	The set of paged query results that this page is from.
	Page.pageRange	query.PageRange (read-only)	Client and server scripts	The range of query results for this page.

## PagedData Object Members

The following members are called on the [query.PagedData](#) object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Method	PagedData.iterator()	Iterator object	Client and server scripts	Standard SuiteScript 2.0 object for iterating through results.
Property	PagedData.count	number (read-only)	Client and server scripts	The total number of paged query results.
	PagedData.pageRanges	query.PageRange[]	Client and server scripts	An array of page ranges for the set of paged query results.
	PagedData.pageSize	number (read-only)	Client and server scripts	The number of query result rows per page.

## PageRange Object Members

The following members are called on the [query.PageRange](#) object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Property	PageRange.index	number (read-only)	Client and server scripts	The array index for this page range.
	PageRange.size	number (read-only)	Client and server scripts	The number of query result rows in this page range.

## Query Object Members

The following members are called on the [query.Query](#) object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Method	Query.and()	query.Condition	Client and server scripts	Creates a new condition (a <a href="#">query.Condition</a> object) that corresponds to a logical conjunction (AND) of the arguments passed to the method. The arguments must be one or more <a href="#">query.Condition</a> objects.
	Query.autoJoin(options)	query.Component	Client and server scripts	Creates a join relationship. After you create the initial query definition, use this method to create your first join. This method selects the correct join type automatically based on the record types that are being joined.
	Query.createColumn(options)	query.Column	Client and server scripts	Creates a query result column based on the <a href="#">query.Query</a> object. Use this method to create columns on the initial query definition created with <a href="#">query.create(options)</a> .
	Query.createCondition(options)	query.Condition	Client and server scripts	Creates a condition (filter column) based on the <a href="#">query.Query</a> object.

Member Type	Name	Return Type/ Value Type	Supported Script Types	Description
				Use this method to create conditions on the initial query definition created with <a href="#">query.create(options)</a> .
	<a href="#">Query.createSort(options)</a>	<a href="#">query.Sort</a>	Client and server scripts	Creates a sort based on the <a href="#">query.Query</a> object. The <a href="#">query.Sort</a> object describes a sort that is placed on a particular query result column or condition.
	<a href="#">Query.join(options)</a>	<a href="#">query.Component</a>	Client and server scripts	Creates a join relationship. This method is an alias to <a href="#">Query.autoJoin(options)</a> .  After you create the initial query definition, use this method, or <a href="#">Query.autoJoin(options)</a> , to create your first join.
	<a href="#">Query.joinFrom(options)</a>	<a href="#">query.Component</a>	Client and server scripts	Creates an explicit directional join relationship from another component to the root component of the search definition (an inverse join). This method sets the <a href="#">Component.source</a> property on the returned <a href="#">query.Component</a> 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.
	<a href="#">Query.joinTo(options)</a>	<a href="#">query.Component</a>	Client and server scripts	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 query types. This method sets the <a href="#">Component.target</a> property on the returned <a href="#">query.Component</a> 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.
	<a href="#">Query.not()</a>	<a href="#">query.Condition</a>	Client and server scripts	Creates a new condition (a <a href="#">query.Condition</a> object) that corresponds to a logical negation (NOT) of the argument passed to the method. The argument must be a <a href="#">query.Condition</a> object.
	<a href="#">Query.or()</a>	<a href="#">query.Condition</a>	Client and server scripts	Creates a new condition (a <a href="#">query.Condition</a> object) that corresponds to a logical disjunction (OR) of the arguments passed to the method. The arguments must be one or more <a href="#">query.Condition</a> objects.
	<a href="#">Query.run()</a>	<a href="#">query.ResultSet</a>	Client and server scripts	Executes the query and returns the query result set.
	<a href="#">Query.run.promise()</a>	<a href="#">query.ResultSet</a>	Client scripts	Executes the query asynchronously and returns the query result set.
	<a href="#">Query.runPaged()</a>	<a href="#">query.PagedData</a>	Client and server scripts	Executes the query and returns a set of paged results.
	<a href="#">Query.runPaged.promise()</a>	<a href="#">query.PagedData</a>	Client scripts	Executes the query asynchronously and returns a set of paged results.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
	<a href="#">Query.toSuiteQL()</a>	<a href="#">query.SuiteQL</a>	Client and server scripts	Converts this <a href="#">query.Query</a> object to its corresponding SuiteQL representation.
Property	<a href="#">Query.child</a>	Object (read-only)	Client and server scripts	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 <a href="#">query.Component</a> object.
	<a href="#">Query.columns</a>	<a href="#">query.Column[]</a>	Client and server scripts	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.
	<a href="#">Query.condition</a>	<a href="#">query.Condition</a> object	Client and server scripts	The parent condition that narrows the query results.  Before you execute the query, you must assign your simple or complex conditions to this property.
	<a href="#">Query.id</a>	number (read-only)	Client and server scripts	The ID of the query definition.  This property has a value only for existing queries that are loaded using <a href="#">query.load(options)</a> . If you create a query using <a href="#">query.create(options)</a> but do not save it, this property is null.
	<a href="#">Query.name</a>	string (read-only)	Client and server scripts	The name of the query definition.  This property has a value only for existing queries that are loaded using <a href="#">query.load(options)</a> . If you create a query using <a href="#">query.create(options)</a> but do not save it, this property is null.
	<a href="#">Query.root</a>	<a href="#">query.Component</a> (read-only)	Client and server scripts	The root component of the query definition.
	<a href="#">Query.sort</a>	<a href="#">query.Column[]</a> (read-only)	Client and server scripts	An array of query result columns used for sorting.
	<a href="#">Query.type</a>	string (read-only)	Client and server scripts	The query type of the initial query definition.

## RelativeDate Object Members

The following members are called on the [query.RelativeDate](#) object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Property	<a href="#">RelativeDate.dateId</a>	string (read-only)	Client and server scripts	The ID of the relative date.
	<a href="#">RelativeDate.end</a>	Object (read-only)	Client and server scripts	The end point of the relative date.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
	RelativeDate.interval	Object (read-only)	Client and server scripts	The interval of the relative date (from the <a href="#">RelativeDate.start</a> point to the <a href="#">RelativeDate.end</a> point).
	RelativeDate.isRange	boolean (read-only)	Client and server scripts	Whether the relative date represents a range of dates or a specific moment in time.
	RelativeDate.start	Object (read-only)	Client and server scripts	The start point of the relative date.
	RelativeDate.value	number (read-only)	Client and server scripts	The value of the relative date.

## Result Object Members

The following members are called on the [query.Result](#) object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Method	Result.asList()	Object	Client and server scripts	The query result as a mapped result.
Property	Result.values	Array<string   number   boolean   null> (read-only)	Client and server scripts	The result values.

## ResultSet Object Members

The following members are called on the [query.ResultSet](#) object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Method	ResultSet.asListedResults()	Object[]	Client and server scripts	Returns a query result set as an array of mapped results.
	ResultSet.iterator()	Iterator object	Client and server scripts	Standard SuiteScript 2.0 object for iterating through results.
Property	ResultSet.columns	<a href="#">query.Column</a> [] (read-only)	Client and server scripts	An array of query result column references.
	ResultSet.results	<a href="#">query.Result</a> [] (read-only)	Client and server scripts	An array of <a href="#">query.Result</a> objects.
	ResultSet.types	string[] (read-only)	Client and server scripts	An array of the return types for <a href="#">ResultSet.results</a> .

## Sort Object Members

The following members are called on the [query.Sort](#) object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Property	Sort.ascending	boolean	Client and server scripts	Whether the sort direction is ascending.
	Sort.caseSensitive	boolean	Client and server scripts	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.
	Sort.column	query.Column (read-only)	Client and server scripts	The query result column that the query results are sorted by.
	Sort.locale	string	Client and server scripts	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.
	Sort.nullsLast	boolean	Client and server scripts	Whether query results with null values are listed at the end of the query results.

## SuiteQL Object Members

The following members are called on the [query.SuiteQL](#) object.

Member Type	Name	Return Type/Value Type	Supported Script Types	Description
Method	SuiteQL.run()	query.ResultSet	Client and server scripts	Runs the SuiteQL query and returns the query results.
	SuiteQL.runPaged (options)	query.PagedData	Client and server scripts	Runs the SuiteQL query as a paged query and returns the paged query results.
Property	SuiteQL.columns	query.Column[]	Client and server scripts	Describes the result columns to be returned from the query.
	SuiteQL.params	Array<string   number   boolean> (read-only)	Client and server scripts	Contains the parameters for the query.
	SuiteQL.query	string (read-only)	Client and server scripts	Holds the string representation of the query.
	SuiteQL.type	string (read-only)	Client and server scripts	Describes the type of the query.  This property uses values from the <a href="#">query.Type</a> enum.

# N/query Module Script Samples

The following script samples demonstrate how to use the features of the N/query module:

- [Sample 1: Query for customer records using joins](#)
- [Sample 2: Query for transaction records using joins](#)
- [Sample 3: Convert a query to SuiteQL and run it](#)
- [Sample 4: Run an arbitrary SuiteQL query](#)

## Sample 1: Query for customer records using joins

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample creates a query for customer records, joins the query with two other query types, and runs the query:

```
/**  
 * @NApiVersion 2.x  
 */  
  
require(['N/query'],  
    function(query) {  
  
        // Create a query definition for customer records  
        var myCustomerQuery = 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 mySalesRepJoin = myCustomerQuery.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 myLocationJoin = mySalesRepJoin.autoJoin({  
            fieldId: 'location'  
        });  
  
        // Create conditions for the query  
        var firstCondition = myCustomerQuery.createCondition({  
            fieldId: 'id',  
            operator: query.Operator.EQUAL,  
            values: 107  
        });  
        var secondCondition = myCustomerQuery.createCondition({  
            fieldId: 'id',  
            operator: query.Operator.EQUAL,  
            values: 108  
        });  
        myCustomerQuery.addCondition(firstCondition);  
        myCustomerQuery.addCondition(secondCondition);  
  
        // Run the query  
        var results = myCustomerQuery.run();  
        var records = results.getResults();  
        var recordCount = records.length;  
        if (recordCount > 0) {  
            for (var i = 0; i < recordCount; i++) {  
                var record = records[i];  
                var salesRepName = record.getValue('salesrep');  
                var locationName = record.getValue('location');  
                log.info('Customer ID: ' + record.getId() + ', Sales Rep: ' + salesRepName + ', Location: ' + locationName);  
            }  
        }  
    }  
);
```

```

        operator: query.Operator.EQUAL,
        values: 2647
    });
    var thirdCondition = mySalesRepJoin.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'.
    myCustomerQuery.condition = myCustomerQuery.and(
        thirdCondition, myCustomerQuery.or(firstCondition, secondCondition)
    );

    // Create query columns
    myCustomerQuery.columns = [
        myCustomerQuery.createColumn({
            fieldId: 'entityid'
        }),
        myCustomerQuery.createColumn({
            fieldId: 'id'
        }),
        mySalesRepJoin.createColumn({
            fieldId: 'entityid'
        }),
        mySalesRepJoin.createColumn({
            fieldId: 'email'
        }),
        mySalesRepJoin.createColumn({
            fieldId: 'hiredate'
        }),
        myLocationJoin.createColumn({
            fieldId: 'name'
        })
    ];
}

// Sort the query results based on query columns
myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[3]
    }),
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[0],
        ascending: false
    })
];
}

// Run the query
var resultSet = myCustomerQuery.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);
});

```

## Sample 2: Query for transaction records using joins

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample creates a query for transaction records, joins the query with another query type, and runs the query as a paged query:

```

/**
 * @NApiVersion 2.x
 */

require(['N/query'],
function(query) {

    // Create a query definition for transaction records
    var myTransactionQuery = 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 myEmployeeJoin = myTransactionQuery.autoJoin({
        fieldId: 'employee'
    });

    // Create a query column
    myTransactionQuery.columns = [
        myEmployeeJoin.createColumn({
            fieldId: 'subsidiary'
        })
    ];

    // Sort the query results based on a query column
    myTransactionQuery.sort = [
        myTransactionQuery.createSort({
            column: myTransactionQuery.columns[0],
            ascending: false
        })
    ];

    // Run the query as a paged query with 10 results per page
    var results = myTransactionQuery.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);
    }
});

```

## Sample 3: Convert a query to SuiteQL and run it

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample creates a query for customer records, converts it to its SuiteQL representation, and runs it:

```

/**
 * @NApiVersion 2.x
 */
require(['N/query'], function(query) {
    var myCustomerQuery = query.create({
        type: query.Type.CUSTOMER
    });

    myCustomerQuery.columns = [
        myCustomerQuery.createColumn({
            fieldId: 'entityid'
        }),
        myCustomerQuery.createColumn({
            fieldId: 'email'
        })
    ];

    myCustomerQuery.condition = myCustomerQuery.createCondition({
        fieldId: 'isperson',
        operator: query.Operator.IS,
        values: [true]
    });
});

```

```

var mySQLCustomerQuery = myCustomerQuery.toSuiteQL();

var results = mySQLCustomerQuery.run();
});

```

## Sample 4: Run an arbitrary SuiteQL query

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample constructs a SuiteQL query string, runs the query as a paged query, and iterates over the results:

```

/**
 * @NApiVersion 2.x
 */
require(['N/query'], function(query) {
    var sql =
        "SELECT " +
        " scriptDeployment.primarykey, scriptexecutioncontextmap.executioncontext " +
        " FROM " +
        " scriptDeployment, scriptexecutioncontextmap " +
        " WHERE " +
        " scriptexecutioncontextmap.scriptrecord = scriptDeployment.primarykey " +
        " AND " +
        " scriptexecutioncontextmap.executioncontext IN ('WEBSTORE', 'WEBAPPLICATION')";

    var resultIterator = query.runSuiteQLPaged({
        query: sql,
        pageSize: 10
    }).iterator();

    resultIterator.each(function(page) {
        var pageIterator = page.value.data.iterator();
        pageIterator.each(function(row) {
            log.debug('ID: ' + row.value.getValue(0) + ', Context: ' + row.value.getValue(1));
            return true;
        });
        return true;
    });
});

```

## 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 any of the following methods to create subsequent joins:
  - [Query.autoJoin\(options\)](#)
  - [Query.joinFrom\(options\)](#)
  - [Query.joinTo\(options\)](#)
  - [Component.autoJoin\(options\)](#)
  - [Component.joinFrom\(options\)](#)
  - [Component.joinTo\(options\)](#)

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 that is relative to the current date. You can use relative dates when you create query conditions.

To create relative dates:

- Use `query.createRelativeDate(options)` to create a `query.RelativeDate` object. When you call `query.createRelativeDate(options)`, use the values in the `query.DateId` enum to specify a date that is 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

Term	Definition	For More Information
Aggregate function	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.	<ul style="list-style-type: none"> <li><a href="#">query.Aggregate</a></li> <li><a href="#">Component.createColumn(options)</a></li> <li><a href="#">Component.createCondition(options)</a></li> <li><a href="#">Query.createColumn(options)</a></li> <li><a href="#">Query.createCondition(options)</a></li> </ul>
Column	A column describes the field types (columns) that are displayed from the query results. A column is also known as a query results column.	<ul style="list-style-type: none"> <li><a href="#">query.Column</a></li> </ul>
Component	When you script queries with the N/query module, your query is made up of one or more components, which are represented as <a href="#">query.Component</a> objects. You can think of a component as a building block; each new component builds on the previous component created. <ul style="list-style-type: none"> <li>The first component created represents the initial search type and is a child of <a href="#">query.Query</a>.</li> </ul>	<ul style="list-style-type: none"> <li><a href="#">query.Component</a></li> </ul>

Term	Definition	For More Information
	<ul style="list-style-type: none"> <li>▪ Each subsequent component created is a child of the previous component.</li> <li>▪ The last component created encapsulates the join relationship between it and all of its parent components.</li> </ul> <p>A query always contains at least one component: the root component. When you create the initial query definition using <a href="#">query.create(options)</a>, 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:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">Query.autoJoin(options)</a>, <a href="#">Query.joinFrom(options)</a>, or <a href="#">Query.joinTo(options)</a></li> <li>▪ <a href="#">Component.autoJoin(options)</a>, <a href="#">Component.joinFrom(options)</a>, or <a href="#">Component.joinTo(options)</a></li> </ul>	
Condition	A condition narrows the query results.	<ul style="list-style-type: none"> <li>▪ <a href="#">query.Condition</a></li> </ul>
Formula	Formulas can be used to create conditions and columns.	<ul style="list-style-type: none"> <li>▪ <a href="#">Formulas in Search</a></li> <li>▪ <a href="#">SQL Expressions</a></li> </ul>
Group	You can summarize your query results into unique groups of column values.	<ul style="list-style-type: none"> <li>▪ <a href="#">Column.groupBy</a></li> </ul>
Join	A join lets you create a query based on a field type that is shared between two record types. You can use <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autojoin(options)</a> to create a join relationship automatically based on a field that you specify. You can use <a href="#">Query.joinFrom(options)/Query.joinTo(options)</a> and <a href="#">Component.joinFrom(options)/Component.joinTo(options)</a> to create explicit directional join relationships from one component to another.	<ul style="list-style-type: none"> <li>▪ <a href="#">query.Query</a></li> <li>▪ <a href="#">query.Component</a></li> </ul>
Page	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.	<ul style="list-style-type: none"> <li>▪ <a href="#">Query.runPaged()</a></li> <li>▪ <a href="#">query.PagedData</a></li> <li>▪ <a href="#">query.PageRange</a></li> <li>▪ <a href="#">query.Page</a></li> </ul>
Paged data	Paged data represents a set of paged query results.	<ul style="list-style-type: none"> <li>▪ <a href="#">Query.runPaged()</a></li> <li>▪ <a href="#">query.PagedData</a></li> <li>▪ <a href="#">query.PageRange</a></li> <li>▪ <a href="#">query.Page</a></li> </ul>
Page range	A page range is a set of pages from a set of paged query results.	<ul style="list-style-type: none"> <li>▪ <a href="#">Query.runPaged()</a></li> <li>▪ <a href="#">query.PagedData</a></li> <li>▪ <a href="#">query.PageRange</a></li> <li>▪ <a href="#">query.Page</a></li> </ul>
Relative date	A relative date is a date that is relative to the current date. You can use relative dates when you create query conditions.	<ul style="list-style-type: none"> <li>▪ <a href="#">query.RelativeDate</a></li> <li>▪ <a href="#">query.createRelativeDate(options)</a></li> <li>▪ <a href="#">query.RelativeDateRange</a></li> </ul>
Result	A result is a single row from a result set.	<ul style="list-style-type: none"> <li>▪ <a href="#">Query.run()</a></li> <li>▪ <a href="#">query.ResultSet</a></li> <li>▪ <a href="#">query.Result</a></li> </ul>
Result set	A result set is a set of query results.	<ul style="list-style-type: none"> <li>▪ <a href="#">Query.run()</a></li> </ul>

Term	Definition	For More Information
		<ul style="list-style-type: none"> <li>■ <a href="#">query.ResultSet</a></li> <li>■ <a href="#">query.Result</a></li> </ul>
Query definition	The query definition is the initial search type you define, plus any subsequent joins you define. The initial query definition is created with <a href="#">query.create(options)</a> .	<ul style="list-style-type: none"> <li>■ <a href="#">query.Query</a></li> </ul>
Query type	The query type is the initial query type of your query definition. It represents the record type you want to query for. It is set with the <a href="#">query.Type</a> enum during the execution of <a href="#">query.create(options)</a> . For example, if you want to query for customer records, specify <code>query.Type.CUSTOMER</code> as the query type when you call <a href="#">query.create(options)</a> .	<ul style="list-style-type: none"> <li>■ <a href="#">query.Query</a></li> <li>■ <a href="#">query.Type</a></li> </ul>
Sort	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).	<ul style="list-style-type: none"> <li>■ <a href="#">query.Sort</a></li> <li>■ <a href="#">Query.createSort(options)</a></li> <li>■ <a href="#">Component.createSort(options)</a></li> </ul>

## 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:

```
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. You can use these operators when working with formulas in SuiteScript or the NetSuite UI. Use the ^ operator to 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. Use the < operator to 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. When you use [Query.autoJoin\(options\)](#) or [Component.autoJoin\(options\)](#), you do not need to use the ^ or < operators to access fields in the joined query.

The following table lists common join operations and the corresponding join trail.

Join Type	Join Operation	Join Trail
Automatic using <a href="#">Query.autoJoin(options)</a> or <a href="#">Component.autoJoin(options)</a>	<pre>// The base record type is Customer var myCustomerQuery = query.create({     type: query.Type.CUSTOMER });  // The joined record type is Employee var mySalesRepJoin = myCustomerQuery.autoJoin({     fieldId: 'salesrep' });</pre>	<p><b>Base record fields (Customer)</b></p> <ul style="list-style-type: none"> <li>▪ <code>customer.&lt;baseFieldName&gt;</code></li> <li>▪ Example: <code>customer.email</code></li> </ul> <p><b>Joined record fields (Employee)</b></p> <ul style="list-style-type: none"> <li>▪ <code>customer.salesrep.&lt;joinedFieldName&gt;</code></li> <li>▪ Example: <code>customer.salesrep.phone</code></li> </ul>
Polymorphic using <a href="#">Query.joinTo(options)</a> or <a href="#">Component.joinTo(options)</a>	<pre>// 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' });</pre>	<p><b>Base record fields (Transaction)</b></p> <ul style="list-style-type: none"> <li>▪ <code>transaction.&lt;baseFieldName&gt;</code></li> <li>▪ Example: <code>transaction.entity</code></li> </ul> <p><b>Joined record fields (Employee)</b></p> <ul style="list-style-type: none"> <li>▪ <code>transaction.&lt;baseFieldName&gt;^employee.&lt;joinedFieldName&gt;</code></li> <li>▪ Example: <code>transaction.createdby^employee.email</code></li> </ul>
Inverse using <a href="#">Query.joinFrom(options)</a> or <a href="#">Component.joinFrom(options)</a>	<pre>// 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' });</pre>	<p><b>Base record fields (Employee)</b></p> <ul style="list-style-type: none"> <li>▪ <code>employee.&lt;baseFieldName&gt;</code></li> <li>▪ Example: <code>employee.entityid</code></li> </ul> <p><b>Joined record fields (Transaction)</b></p> <ul style="list-style-type: none"> <li>▪ <code>employee.&lt;baseFieldName&gt;&lt;transaction.daysoverdue&gt;</code></li> <li>▪ Example: <code>employee.entity&lt;transaction.daysoverdue&gt;</code></li> </ul>

## 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:

```
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. The `query.RelativeDate` object represents a specific date or moment in time relative to the current date. Each of the values in the `query.RelativeDateRange` enum represents a range of dates relative to the current date. 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 relative date 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 a date two days before the current date:

```
var myDatesAgo = query.createRelativeDate({
  dateId: query.DateId.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:

```
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 a date two days after the current date:

```
var myDatesFromNow = query.createRelativeDate({
  dateId: query.DateId.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 object represents a specific moment in time, so the current date is used automatically as one of the boundaries. When you use a single `query.RelativeDateRange` enum value, the enum value represents a range of dates, so the `start` and `end` properties of the date range are used automatically as the boundaries. For example:

```
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:

```
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:

```
var myEndDate = query.createRelativeDate({
  dateId: query.DateId.WEEKS_AGO,
  value: 2
});
```

```
var myComplexCondition = myQuery.createCondition({
    fieldId: 'trandate',
    operator: query.Operator.WITHIN,
    values: [query.RelativeDateRange.THREE_FISCAL_YEARS_AGO.start, myEndDate]
});
```

## SuiteQL in the N/query Module

SuiteQL is a query language based on the SQL-92 revision of the SQL database query language. It provides advanced query capabilities you can use to access your NetSuite records and data. For more information about SuiteQL, including limitations, exceptions, and more usage examples, see the help topic [SuiteQL](#).

In SuiteScript, you can create and run SuiteQL queries using the N/query module. Queries created using SuiteQL can be more powerful and flexible than queries created using other APIs in the N/query module. SuiteQL queries can also provide the best query performance for many use cases. You can create your own SuiteQL query strings, which lets you design and run complex SQL queries that cannot be created otherwise. For examples of SuiteQL queries you can create, see [Examples of Using SuiteQL in the N/query Module](#).



**Important:** To create your own SuiteQL query strings, you must know the names of the record types and fields you want to use. For more information about finding record type and field names, see the help topic [Finding Record Type and Field Names](#).

Using the N/query module, you can run SuiteQL queries in the following ways:

- Convert an existing query (as a `query.Query` object) to its SuiteQL representation (as a `query.SuiteQL` object) and run the query. To learn more, see [Converting an Existing Query to SuiteQL](#).
- Run an arbitrary SuiteQL query as a paged or non-paged query. To learn more, see [Running an Arbitrary SuiteQL Query](#).

## Converting an Existing Query to SuiteQL

If you already have a `query.Query` object in your script (one that you created using `query.create(options)` or loaded using `query.load(options)`), you can convert the query to its SuiteQL representation. The `Query.toSuiteQL()` method converts a `query.Query` object to a `query.SuiteQL` object. The resulting `query.SuiteQL` object represents the same query as the original `query.Query` object and, when run, returns the same query results.



**Important:** The resulting SuiteQL query string (contained in the `SuiteQL.query` property) does not include any aliases you set on query result columns in the original `query.Query` object. For more information about aliases, see [Column.alias](#).

A `query.SuiteQL` object includes the following properties:

- `SuiteQL.columns` — The result columns to be returned from the query. This property is an array of `query.Column` objects.
- `SuiteQL.params` — The parameters for the query. In SuiteQL, query conditions are represented using the `WHERE` clause and a set of parameters.
- `SuiteQL.query` — The string representation of the SuiteQL query. This string can contain SQL clauses, record or table names, field names, operators, and more.
- `SuiteQL.type` — The type of the query. This property uses values from the `query.Type` enum.

To run the SuiteQL query, use one of the following methods:

- Use [SuiteQL.run\(\)](#) to run the query as a non-paged query. This method returns the results as a [query.ResultSet](#) object.
- Use [SuiteQL.runPaged\(options\)](#) to run the query as a paged query. This method returns the results as a [query.PagedData](#) object. The default page size is 50 results per page.

The following example shows you how to create a query as a [query.Query](#) object, convert the query to SuiteQL, and run the resulting SuiteQL query as a non-paged query:

```
var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'email'
    })
];

myCustomerQuery.condition = myCustomerQuery.createCondition({
    fieldId: 'isperson',
    operator: query.Operator.IS,
    values: [true]
});

var mySQLCustomerQuery = myCustomerQuery.toSuiteQL();

var results = mySQLCustomerQuery.run();
```

In the preceding example, the `mySQLCustomerQuery` variable contains the resulting [query.SuiteQL](#) object after the conversion. In this object, the [SuiteQL.query](#) property contains the string representation of the original query. In this example, this property contains the following string:

```
SELECT customer.entityid AS entityidRAW /*{entityid#RAW}*/, customer.email AS emailRAW /*{email#RAW}*/ FROM
customer WHERE customer.isperson = ?
```

The [SuiteQL.params](#) property contains the parameter value `true`. When the SuiteQL query runs, the question mark (?) in the query string is replaced with the parameter value (`true`).

## Running an Arbitrary SuiteQL Query

You can design your own SuiteQL queries and run them. The [query.runSuiteQL\(options\)](#) method lets you run an arbitrary SuiteQL query, and it returns query results as a [query.ResultSet](#) object. The [query.runSuiteQLPaged\(options\)](#) method lets you run a SuiteQL query as a paged query, and it returns query results as a [query.PagedData](#) object.

To specify the SuiteQL query to run, you can provide one of the following to [query.runSuiteQL\(options\)](#) or [query.runSuiteQLPaged\(options\)](#):

- A string representation of the SuiteQL query

```
var results = query.runSuiteQL({
```

```
    query: 'SELECT customer.entityid, customer.email FROM customer'
  });
}
```

- A `query.SuiteQL` object

```
// In this example, mySuiteQLCustomerQuery is an existing SuiteQL object
var results = query.runSuiteQL(mySuiteQLCustomerQuery);
```

- A JavaScript Object that contains a `query` property and, optionally, a `params` property:

```
var results = query.runSuiteQL({
  query: 'SELECT customer.entityid, customer.email FROM customer WHERE customer.isperson = ?',
  params: [true]
});
```

## Examples of Using SuiteQL in the N/query Module

The following examples demonstrate how to create and run SuiteQL queries using the features in the N/query module.

### Convert an Existing Query to SuiteQL

This example RESTlet loads a SuiteAnalytics Workbook with an ID of `OpenSalesOrders` and runs it. The RESTlet then converts the query to SuiteQL and runs it. The RESTlet returns both sets of query results.

```
/**
 * @NApiVersion 2.x
 * @NScriptType restlet
 */
define(['N/query'], function(query) {
  return {
    get: function(context) {
      // Load the workbook by name (record ID)
      var openSalesOrders = query.load('OpenSalesOrders');

      // Run the query
      var resultQuery = openSalesOrders.run();

      // Convert the query to its SuiteQL representation
      var openSalesOrdersQL = openSalesOrders.toSuiteQL();

      // Examine the SuiteQL query string
      var suiteQL = openSalesOrdersQL.query;

      // Run the SuiteQL query
      var resultSuiteQL = query.runSuiteQL(suiteQL);

      // Compose the RESTlet response
      var response = {
        query: openSalesOrders,
        resultQuery: resultQuery,
        suiteQL: suiteQL,
        resultSuiteQL: resultSuiteQL
      };
    }
  };
});
```

```

        // Return the response
        return JSON.stringify(response);
    }
}
});

```

## Create a Custom SuiteQL Query

This example RESTlet constructs a custom query string using SuiteQL, runs the query, and returns the query results.

```

/**
 * @NApiVersion 2.x
 * @NScriptType restlet
 */
define(['N/query'], function(query) {
    return {
        get: function(context) {
            // Construct the SuiteQL query string
            var suiteQL =
"SELECT "+

" \"TRANSACTION\".tranid AS tranidRAW /*{tranid#RAW}*/,  "+
" \"TRANSACTION\".trandate AS trandateRAW /*{trandate#RAW}*/,  "+
" \"TRANSACTION\".postingperiod AS postingperiodDISPLAY /*{postingperiod#DISPLAY}*/,  "+
" BUILTIN.DF(\"TRANSACTION\".postingperiod) AS postingperiodDISPLAY /*{postingperiod#DISPLAY}*/,  "+
" BUILTIN.DF(\"TRANSACTION\".status) AS statusDISPLAY /*{status#DISPLAY}*/,  "+
" BUILTIN.DF(transactionLine.item) AS transactionlinesitemDISPLAY /*{transactionlines.item#DISPLAY}*/,  "+
" BUILTIN.DF(\"TRANSACTION\".entity) AS entityDISPLAY /*{entity#DISPLAY}*/,  "+
" transactionLine.quantity * -1 AS transactionlinesquantity /*-{transactionlines.quantity}*/,  "+
" BUILTIN.CONSOLIDATE(transactionLine.netamount, 'LEDGER', 'DEFAULT', 'DEFAULT', 1, 396, 'DEFAULT') AS
transactionlinesnetamountCU /*{transactionlines.netamount#CURRENCY_CONSOLIDATED}*/,  "+
" BUILTIN.CURRENCY(BUILTIN.CONSOLIDATE(transactionLine.netamou
nt, 'LEDGER', 'DEFAULT', 'DEFAULT', 1, 396, 'DEFAULT')) AS transactionlinesnetamountCU_C /*{transactionlines.
netamount#CURRENCY_CONSOLIDATED}*/,  "+
" CUSTOMRECORD41.custrecord14 AS custrecord15customrecord41c /*{custrecord15<customrecord41.custrecord14#RAW}*/+
/* "FROM " + " \"TRANSACTION\"", "+ " CUSTOMRECORD41, "+ " \"ACCOUNT\"", "+ " TransactionAccountingLine, "+

" transactionLine "+ " WHERE "+ " (((\"TRANSACTION\".\"ID\" = CUSTOMRECORD41.custrecord15 AND TransactionA
ccountingLine.\"ACCOUNT\" = \"ACCOUNT\".\"ID\"(+)) AND (transactionLine.\"TRANSACTION\" = TransactionAccounti
ngLine.\"TRANSACTION\" AND transactionLine.\"ID\" = TransactionAccountingLine.transactionline)) AND \"TRANSACTI
ON\".\"ID\" = transactionLine.\"TRANSACTION\")) "+ " AND ((UPPER(\"TRANSACTION\".\"TYPE\") IN ('SALESORD') AND
UPPER(\"TRANSACTION\".status) IN ('SALESORD:D', 'SALESORD:E', 'SALESORD:B') AND UPPER(\"ACCOUNT\".accttype) IN
('INCOME') AND (NOT( "+ " transactionLine.itemtype IN ('ShipItem') "+ " ) OR transactionLine.itemtype IS NULL)
AND ((transactionLine.quantity * -1) - NVL(transactionLine.quantitycommitted, 0)) - NVL(transactionLine.quantit
yshiprev, 0)> 0 AND NVL(transactionLine.mainline, 'F') = 'F' AND NVL(transactionLine.taxline, 'F') = 'F' AND
NVL(transactionLine.isclosed, 'F') = 'F')) "+

"ORDER BY "+

" \"TRANSACTION\".trandate ASC NULLS LAST";

        // Run the SuiteQL query
        var resultSuiteQL = query.runSuiteQL(suiteQL);

        // Compose the RESTlet response
        var response = {
            resultSuiteQL: resultSuiteQL
        };
    }
}
);

```

```

    };

    // Return the response
    return JSON.stringify(response);
}
}
});

```

## Accept a SuiteQL Query as an Argument

This example RESTlet accepts a SuiteQL query as a provided argument to the `get` endpoint, runs the query, and returns the query results.

```

/**
 * @NApiVersion 2.x
 * @NScriptType restlet
 */
define(['N/query'], function(query) {
    return {
        get: function(context) {
            return runQuery(query, context);
        },
        post: function(context) {
            return runQuery(query, context);
        }
    }

    function runQuery(query, context) {
        // Get the SuiteQL query string from the arguments. For example, the
        // SuiteQL query may look like the following:
        //
        // &suiteQL=select%20type%2C%20BUILTIN.DF(type)%2C%20tranid%2C%20trandate%20from%20transaction%20where%
        20type%3D'SalesOrd'
        var id = context.id;
        var suiteQL = context.suiteQL;

        // Run the query
        var resultSuiteQL = query.runSuiteQL(suiteQL);

        // Compose the RESTlet response
        var response = {
            id: id,
            suiteQL: suiteQL,
            resultSuiteQL: resultSuiteQL
        };

        // Return the response
        return JSON.stringify(response);
    };
});

```

## Run a Paged SuiteQL Query

This example script runs a SuiteQL query as a paged query with a page size of 10 results per page.

```

/**
 * @NApiVersion 2.x
 */
require(['N/query'], function(query) {
    // Construct the SuiteQL query string
    var sql =
        "SELECT " +
        " scriptDeployment.primarykey, scriptexecutioncontextmap.executioncontext " +
        " FROM " +
        " scriptDeployment, scriptexecutioncontextmap " +
        " WHERE " +
        " scriptexecutioncontextmap.scriptrecord = scriptDeployment.primarykey " +
        " AND " +
        " scriptexecutioncontextmap.executioncontext IN ('WEBSTORE', 'WEBAPPLICATION')";

    // Run the SuiteQL query as a paged query and return an iterator
    var resultIterator = query.runSuiteQLPaged({
        query: sql,
        pageSize: 10
    }).iterator();

    // Use the iterator to process each page of results
    resultIterator.each(function(page) {
        var pageIterator = page.value.data.iterator();
        pageIterator.each(function(row) {
            log.debug('ID: ' + row.value.getValue(0) + ', Context: ' + row.value.getValue(1));
            return true;
        });

        return true;
    });
});

```

## Use a SuiteQL Query in a Map/Reduce Script

This example map/reduce script uses a SuiteQL query as the source of input data. The value 271 is the internal ID of a transaction record.

```

/**
 * @NApiVersion 2.x
 * @NScriptType mapreducescript
 */
define(['N/query'], function(query) {
    // Define the getInputData() stage function
    function getInputData() {
        // Construct the SuiteQL query string
        var suiteQL =
            "SELECT " +
            " \"TRANSACTION\".tranid AS tranidRAW /*{tranid#RAW}*/, " +
            " \"TRANSACTION\".trandate AS trandateRAW /*{trandate#RAW}*/, " +
            " \"TRANSACTION\".postingperiod AS postingperiodDISPLAY /*{postingperiod#DISPLAY}*/ " +
            "FROM " +
            " \"TRANSACTION\" WHERE \"TRANSACTION\".\"ID\" = ? ";

```

```

// Return the query results as input data. The value 271 is the
// internal ID of a transaction record
return {
    type: 'suiteql',
    query: suiteQL,
    params: [271]
};

// Define the map() stage function
function map(context) {
    context.write(context.key, context.value);
}

// Define the reduce() stage function
function reduce(context) {
    context.write(context.key, context.values[0]);
}

// Define the summarize() stage function
function summarize(summary) {
    if (summary.inputSummary.error) {
        log.debug('An error occurred.');
    }
}

// Return the function definitions
return {
    getInputData: getInputData,
    map: map,
    reduce: reduce,
    summarize: summarize
}
});

```

## query.Column

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	<p>A query result column.</p> <p>The <code>query.Column</code> object is the equivalent of the <code>search.Column</code> object in the <a href="#">N/search Module</a>. The <code>query.Column</code> object describes the field types (columns) that are displayed from the query results.</p> <p>To create columns:</p> <ul style="list-style-type: none"> <li>■ Use <code>Query.createColumn(options)</code> to create a column on the initial query definition created with <code>query.create(options)</code>.</li> <li>■ Use <code>Component.createColumn(options)</code> to create a column on a join relationship created with <code>Query.autoJoin(options)</code> or <code>Component.autoJoin(options)</code>.</li> <li>■ Assign all created columns as array values to <code>Query.columns</code>. For an example, see <a href="#">Syntax</a>.</li> </ul>
<b>Supported Script Types</b>	Client and server scripts

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/query Module
<b>Methods and Properties</b>	<a href="#">Column Object Members</a>
<b>Since</b>	2018.1

## Syntax

**Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'id'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'entityid'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'email'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'hiredate'
    }),
];
;

myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[1]
    }),
    mySalesRepJoin.createSort({
        column: mySalesRepJoin.columns[0],
        ascending: false
    })
];
;

var resultSet = myCustomerQuery.run();
...
// Add additional code
```

## Column.aggregate



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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.  This property is set when <a href="#">Query.createColumn(options)</a> or <a href="#">Component.createColumn(options)</a> is executed.  For a list of supported aggregate functions, see the <a href="#">query.Aggregate</a> enum.
<b>Type</b>	string (read-only)
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Column</a>
<b>Sibling Object Members</b>	<a href="#">Column Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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 myAggColumn = myTransactionQuery.createColumn({
    fieldId: 'amount',
    aggregate: query.Aggregate.AVERAGE
});

myTransactionQuery.columns = [myAggColumn];

var theAggregate = myAggColumn.aggregate;
...
// Add additional code
```

## Column.alias



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	An alias for this column. An alias is an alternate name for a column, and the alias is used in mapped results.  In general, the alias is an optional property. If you want to use mapped results in your script, the alias is required. To use mapped results, you must specify an alias in the following situations: <ul style="list-style-type: none"> <li>■ You must specify an alias for a column when the column uses a formula.</li> </ul>
-----------------------------	--

	<ul style="list-style-type: none"> <li>You must specify an alias when two columns in a joined query use the same field ID. For example, many record types include the <b>entity</b> field ID. If you join two record types that use the <b>entity</b> field ID, and you use the <b>entity</b> field ID to create result columns for both record types, you must specify an alias for one of the columns. This alias distinguishes the two columns that have the same field ID.</li> </ul> <p>This property is set when <a href="#">Query.createColumn(options)</a> or <a href="#">Component.createColumn(options)</a> is executed.</p>
<b>Type</b>	string (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Column
<b>Sibling Object Members</b>	<a href="#">Column Object Members</a>
<b>Since</b>	2019.2

## Syntax



**Important:** The following code sample 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 myAliasColumn = myTransactionQuery.createColumn({
    fieldId: 'amount',
    aggregate: query.Aggregate.AVERAGE,
    alias: 'sunkcostamount'
});

myTransactionQuery.columns = [myAliasColumn];

var theAlias = myAliasColumn.alias;
...
// Add additional code
```

## Column.component



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	A reference to the <a href="#">query.Component</a> object to which this query result column belongs.  This property is set when <a href="#">Query.createColumn(options)</a> or <a href="#">Component.createColumn(options)</a> is executed.
<b>Type</b>	<a href="#">query.Component</a> (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Column

Sibling Object Members	<a href="#">Column Object Members</a>
Since	2018.1

## Syntax

 **Important:** The following code sample 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 theComponent = myAmountColumn.component;
...
// Add additional code
```

## Column.context

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The field context for values in the query result column.  This property is set when <a href="#">Query.createColumn(options)</a> or <a href="#">Component.createColumn(options)</a> is executed. The field context determines how field values are displayed in a column. For example, you can specify that a column should display raw data (such as internal IDs), consolidated or converted amounts (such as currency totals), or user-friendly values (such as names).  This property is an Object that includes the name of the context (which is a value in the <a href="#">query.FieldContext</a> enum) and any parameters that apply to that context. In this release, only the <a href="#">query.FieldContext.CONVERTED</a> context uses parameters. For information about these parameters, see <a href="#">Query.createColumn(options)</a> or <a href="#">Component.createColumn(options)</a> .
<b>Type</b>	Object (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.Column</a>
<b>Sibling Object Members</b>	<a href="#">Column Object Members</a>
<b>Since</b>	2019.1

## Syntax

 **Important:** The following code sample 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 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',
        context: query.FieldContext.CURRENCY_CONSOLIDATED
    })
];

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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The name of the query result column.  This property is set during the execution of <a href="#">Query.createColumn(options)</a> or <a href="#">Component.createColumn(options)</a> . This property and the <a href="#">Column.formula</a> property cannot be set at the same time.
<b>Type</b>	string (read-only)
<b>Module</b>	<a href="#">N/query Module</a>

<b>Parent Object</b>	query.Column
<b>Sibling Object Members</b>	Column Object Members
<b>Since</b>	2018.1

## Syntax

**Important:** The following code sample 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

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	A formula used to create the query result column.  This property is set during the execution of <a href="#">Query.createColumn(options)</a> or <a href="#">Component.createColumn(options)</a> . This property and the <a href="#">Column.fieldId</a> property cannot be set at the same time.  For more information on formulas, see the help topics <a href="#">Formulas in Search</a> and <a href="#">SQL Expressions</a> .
<b>Type</b>	string (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Column
<b>Sibling Object Members</b>	Column Object Members
<b>Since</b>	2018.1

## Syntax

**Important:** The following code sample 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 theFormula = myFormulaColumn.formula;
...
// Add additional code

```

## Column.groupBy



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Whether the query results are grouped by this query result column. This property is set during the execution of <a href="#">Query.createColumn(options)</a> or <a href="#">Component.createColumn(options)</a> .
<b>Type</b>	boolean (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.Column</a>
<b>Sibling Object Members</b>	<a href="#">Column Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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
});

var myGroupByColumn = myCustomerQuery.createColumn({
    fieldId: 'currency',
    groupBy: true
});

var theGroupBy = myGroupByColumn.groupBy;
...
// Add additional code

```

## Column.type



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The return type of the formula used to create the query result column.  This property is set during the execution of <a href="#">Query.createColumn(options)</a> or <a href="#">Component.createColumn(options)</a> . 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.  For more information on formulas, see the help topics <a href="#">Formulas in Search</a> and <a href="#">SQL Expressions</a> .
<b>Type</b>	string (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Column
<b>Sibling Object Members</b>	<a href="#">Column Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	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 ( <a href="#">query.Query</a> ).
---------------------------	--

	<p>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.</p> <p>The query definition always contains at least one component. Queries with joins contain multiple components. The query definition (<code>query.Query</code>) contains a child <code>query.Component</code> object for each of the following:</p> <ul style="list-style-type: none"> <li>■ <b>The initial query definition:</b> The initial <code>query.Component</code> object is called the root component. It encapsulates the initial query type passed to <code>query.create(options)</code>. The root component is automatically created with the <code>query.Query</code> object and is a child of the <code>query.Query</code> object. The <code>Query.root</code> property contains a reference to the root component.</li> <li>■ <b>The first join:</b> The second <code>query.Component</code> object is created with <code>Query.autoJoin(options)</code>. It encapsulates the relationship between the initial query definition and the second query type. This relationship is determined by the join ID passed to <code>Query.autoJoin(options)</code>. The second <code>query.Component</code> object is a child of the root component.</li> <li>■ <b>Each subsequent join:</b> The third <code>query.Component</code> object is created with <code>Component.autoJoin(options)</code>. All subsequent joins and their respective <code>query.Component</code> objects are also created with <code>Component.autoJoin(options)</code>. Each of these <code>query.Component</code> objects encapsulates the relationship between all previous query types and the new query type. This relationship is determined by the join ID passed to <code>Component.autoJoin(options)</code>.</li> </ul>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/query Module
<b>Methods and Properties</b>	<a href="#">Component Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'id'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'joinid'
    })
];
```

```

        fieldId: 'entityid'
    }),
mySalesRepJoin.createColumn({
    fieldId: 'email'
}),
mySalesRepJoin.createColumn({
    fieldId: 'hiredate'
}),
];
myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[1]
    }),
    mySalesRepJoin.createSort({
        column: mySalesRepJoin.columns[0],
        ascending: false
    })
];
var resultSet = myCustomerQuery.run();
...
// Add additional code

```

## Component.autoJoin(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates a join relationship.</p> <p>Use the method <a href="#">query.create(options)</a> to create your initial query definition (<a href="#">query.Query</a>). The initial query definition uses one query type. For available query types, see <a href="#">query.Type</a>.</p> <p>After you create the initial query definition, use <a href="#">Query.autoJoin(options)</a> to create your first join (<a href="#">query.Component</a>). Then use <a href="#">Component.autoJoin(options)</a> to create each subsequent join (<a href="#">query.Component</a>).</p> <div style="border: 1px solid #f0c080; padding: 5px; margin-top: 10px;"> <b>Important:</b> The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic <a href="#">Available Record Types</a>.       </div>
<b>Returns</b>	<a href="#">query.Component</a>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.Component</a>
<b>Sibling Object Members</b>	<a href="#">Component Object Members</a>
<b>Since</b>	2018.2

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.fieldId</code>	string	required	<p>The column type (field type) that joins the parent component to the new component.</p> <p>Obtain this value from the <a href="#">Records Browser</a>:</p> <ol style="list-style-type: none"> <li>1. Go to the parent component's record type.</li> <li>2. Scroll until you see the Search Joins table.</li> <li>3. Locate the appropriate value in the Join ID column.</li> </ol> <p>For more information on the Records Browser, see <a href="#">Using the SuiteScript Records Browser</a>.</p>

## Errors

Error Code	Thrown If
<code>RELATIONSHIP_ALREADY_USED</code>	The specified join relationship already exists.

## Syntax

 **Important:** The following code sample 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 myEntityJoin = myTransactionQuery.autoJoin({
    fieldId: 'entity'
});

myTransactionQuery.columns = [
    myEntityJoin.createColumn({
        fieldId: 'subsidiary'
    })
];

myTransactionQuery.sort = [
    myTransactionQuery.createSort({
        column: myTransactionQuery.columns[0],
        ascending: false
    })
];
```

```
var results = myTransactionQuery.runPaged({
  pageSize: 10
});
...
// Add additional code
```

## Component.createColumn(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates a query result column based on the <a href="#">query.Component</a> object. The <a href="#">query.Column</a> object is the equivalent of the <a href="#">search.Column</a> object in the <a href="#">N/search Module</a>. The <a href="#">query.Column</a> object describes the field types (columns) that are displayed from the query results.</p> <p>To create columns:</p> <ul style="list-style-type: none"> <li>▪ Use <a href="#">Component.createColumn(options)</a> to create columns on the join relationships created with <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autoJoin(options)</a>. Use this method in one of two ways:           <ul style="list-style-type: none"> <li>▫ Pass in an argument for the parameter <code>options.fieldId</code>.</li> <li>▫ Pass in an argument for the parameter <code>options.formula</code>. If you use this option, you can also use the optional parameter <code>options.type</code>.</li> </ul> </li> <li>▪ If needed, use <a href="#">Query.createColumn(options)</a> to create columns on the initial query definition created with <a href="#">query.create(options)</a>.</li> <li>▪ Assign all created columns as array values to <a href="#">Query.columns</a>. For an example, see <a href="#">Syntax</a>.</li> </ul> <p>When you create a column, you can specify a field context. The field context determines how field values are displayed in the column. For example, you can specify that a column should display raw data (such as internal IDs), consolidated or converted amounts (such as currency totals), or user-friendly values (such as names). You can specify a field context in two ways:</p> <ul style="list-style-type: none"> <li>▪ Use a context from the <a href="#">query.FieldContext</a> enum directly as the value of the <code>options.context</code> parameter. For example:</li> </ul> <pre><code>myTransactionLine.createColumn({   fieldId: 'netamount',   context: query.FieldContext.CURRENCY_CONSOLIDATED });</code></pre> <p>This example is the simplest way to specify a field context that does not accept additional parameters. Because the <code>options.context</code> parameter is an Object, this example is equivalent to the following:</p> <pre><code>myTransactionLine.createColumn({   fieldId: 'netamount',   context: {     name: query.FieldContext.CURRENCY_CONSOLIDATED   } });</code></pre> <ul style="list-style-type: none"> <li>▪ Use a context from the <a href="#">query.FieldContext</a> enum as the value of the <code>options.context.name</code> parameter, and specify additional parameters using the <code>options.context.params</code> parameter. For example:</li> </ul> <pre><code>myTransactionLine.createColumn({</code></pre>
---------------------------	---

	<pre>         fieldId: 'netamount',         context: {             name: query.FieldContext.CONVERTED,             params: {                 currencyId: 4,                 date: new Date('2019/01/01')             }         }     }); </pre>
	<p>In this release, only the <code>query.FieldContext.CONVERTED</code> context uses additional parameters. The supported parameters are <code>currencyId</code> and <code>date</code>. For the <code>date</code> parameter, you can pass a JavaScript <code>Date</code> object or <code>query.RelativeDate</code> object.</p>
Returns	<code>query.Column</code>
Supported Script Types	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Governance	None
Module	N/query Module
Parent Object	<code>query.Component</code>
Sibling Object Members	Component Object Members
Since	2018.1

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.fieldId</code>	string	required if <code>options.formula</code> is not used	<p>The name of the query result column. This value sets the <code>Column.fieldId</code> property.</p> <p>Obtain this value from the <a href="#">Records Browser</a>:</p> <ol style="list-style-type: none"> <li>1. Go to the appropriate record type.</li> <li>2. Scroll until you see the Search Columns table.</li> <li>3. Locate the appropriate value in the Internal ID column.</li> </ol> <p>For more information on the Records Browser, see <a href="#">Using the SuiteScript Records Browser</a>.</p>
<code>options.formula</code>	string	required if <code>options.fieldId</code> is not used	<p>The formula used to create the query result column. This value sets the <code>Column.formula</code> property.</p> <p>For more information on formulas, see the help topics <a href="#">Formulas in Search</a> and <a href="#">SQL Expressions</a>.</p>
<code>options.type</code>	string	required if <code>options.formula</code> is used	If you use the <code>options.formula</code> 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 <code>Column.type</code> property.

Parameter	Type	Required / Optional	Description
			Use the appropriate <a href="#">query.ReturnType</a> enum value to pass in your argument. This enum holds all the supported values for this parameter.
options.aggregate	string	optional	<p>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 <a href="#">Column.aggregate</a> property.</p> <p>Use the appropriate <a href="#">query.Aggregate</a> enum value to pass in your argument. This enum holds all the supported values for this parameter.</p>
options.groupBy	boolean	optional	<p>Whether the query results are grouped by this query result column. This value sets the <a href="#">Column.groupBy</a> property.</p> <p>If you do not pass in an argument, the default value is set to <code>false</code>.</p>
options.context	Object	optional	The field context for values in the query result column. This value sets the <a href="#">Column.context</a> property.
options.context.name	string	required if <a href="#">options.context</a> is used	<p>The name of the field context.</p> <p>Use the appropriate <a href="#">query.FieldContext</a> enum value to pass in your argument. This enum holds all the supported values for this parameter.</p>
options.context.params	Object	required if <a href="#">options.context.name</a> has a value of <a href="#">query.FieldContext.CONVERTED</a>	<p>The additional parameters to use with the specified field context.</p> <p>In this release, only the <a href="#">query.FieldContext.CONVERTED</a> context uses additional parameters. The supported parameters are <code>currencyId</code> and <code>date</code>.</p>
options.context.params.currencyId	number	required if <a href="#">options.context.name</a> has a value of <a href="#">query.FieldContext.CONVERTED</a>	<p>The internal ID of the currency to convert to.</p> <p>You can specify the internal ID of any currency that is configured in your NetSuite account. For more information, see the help topic <a href="#">Multiple Currencies</a>.</p>
options.context.params.date	<a href="#">query.RelativeDate</a>   <a href="#">JavaScript Date</a>	required if <a href="#">options.context.name</a> has a value of <a href="#">query.FieldContext.CONVERTED</a>	<p>The date to use for the actual exchange rate between the base currency and the currency to convert to.</p> <p>For example, if you want to use the exchange rate that was in effect on March 3, 2019, specify a <a href="#">query.RelativeDate</a> object or <a href="#">JavaScript Date</a> object that represents this date.</p>

## Syntax



**Important:** The following code sample 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
});
```

```

var mySalesRepJoin = myCustomerQuery.join({
    fieldId: 'salesrep'
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'id'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'entityid'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'email'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'hiredate'
    })
];

```

```

myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[1]
    }),
    mySalesRepJoin.createSort({
        column: mySalesRepJoin.columns[0],
        ascending: false
    })
];

```

```

var resultSet = myCustomerQuery.run();
...
// Add additional code

```

## Component.createCondition(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates a condition (query filter) based on the <a href="#">query.Component</a> object.</p> <p>A condition narrows the query results. The <a href="#">query.Condition</a> object acts in the same capacity as the <a href="#">search.Filter</a> object in the <a href="#">N/search Module</a>. The primary difference is that <a href="#">query.Condition</a> objects can contain other <a href="#">query.Condition</a> objects.</p> <p>To create conditions:</p> <ul style="list-style-type: none"> <li>■ Use <a href="#">Component.createCondition(options)</a> to create conditions on the join relationships created with <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autoJoin(options)</a>. Use this method in one of two ways: <ul style="list-style-type: none"> <li>□ Pass in arguments for the parameters <code>options.fieldId</code>, <code>options.operator</code>, and <code>options.values</code>. The combination of these arguments translates to <code>&lt;filter column&gt;&lt;operator&gt;&lt;field value&gt;</code> (for example, 'city' equals 'Boston').</li> </ul> </li> </ul>
---------------------------	--

	<ul style="list-style-type: none"> <li>□ Pass in an argument for the parameter <code>options.formula</code>. If you use this option, you can also use the optional parameter <code>options.type</code>.</li> <li>■ If needed, use <code>Query.createCondition(options)</code> to create conditions on the initial query definition created with <code>query.create(options)</code>.</li> <li>■ If you have multiple conditions, use them to create a new nested condition with the methods <code>Query.and()</code>, <code>Query.or()</code>, and <code>Query.not()</code>.</li> <li>■ Assign your simple or nested condition to <code>Query.condition</code>. For an example, see <a href="#">Syntax</a>.</li> </ul>
Returns	<code>query.Condition</code>
Supported Script Types	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Governance	None
Module	<a href="#">N/query Module</a>
Parent Object	<code>query.Component</code>
Sibling Object Members	Component Object Members
Since	2018.1

## Parameters



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

Parameter	Type	Required / Optional	Description
<code>options.fieldId</code>	string	required if <code>options.operator</code> and <code>options.values</code> are used	<p>The name of the condition. This value sets the <code>Condition.fieldId</code> property.</p> <p>Obtain this value from the <a href="#">Records Browser</a>:</p> <ol style="list-style-type: none"> <li>1. Go to the appropriate record type.</li> <li>2. Scroll until you see the Search Filters table.</li> <li>3. Locate the appropriate value in the Internal ID column.</li> </ol> <p>For more information on the Records Browser, see <a href="#">Using the SuiteScript Records Browser</a>.</p>
<code>options.operator</code>	string	required if <code>options.fieldId</code> and <code>options.values</code> are used	<p>The operator used by the condition. This value sets the <code>Condition.operator</code> parameter.</p> <p>Use the appropriate <code>query.Operator</code> enum value to pass in your argument. This enum holds all the supported values for this parameter.</p>
<code>options.values</code>	<code>string[]   Date[]</code>	required if <code>options.fieldId</code> and <code>options.operator</code> are used, and <code>options.operator</code>	An array of values to use for the condition. This value sets the <code>Condition.values</code> property.

Parameter	Type	Required / Optional	Description
		does <b>not</b> have a value of <code>query.Operator.EMPTY</code> or <code>query.Operator.EMPTY_NOT</code>	
<code>options.formula</code>	string	required if <code>options.fieldId</code> and <code>options.operator</code> are <b>not</b> used	The formula used to create the condition. This value sets the <a href="#">Condition.formula</a> property.  For more information on formulas, see the help topics <a href="#">Formulas in Search</a> and <a href="#">SQL Expressions</a> .
<code>options.type</code>	string	required if <code>options.formula</code> is used	If you use the <code>options.formula</code> 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 <a href="#">Condition.type</a> property.  Use the appropriate <code>query.ReturnType</code> enum value to pass in your argument. This enum holds all the supported values for this parameter.
<code>options.aggregate</code>	string	optional	An aggregate function to run on the condition. An aggregate function performs a calculation on the condition values and returns a single value. This value sets the <a href="#">Condition.aggregate</a> property.  Use the appropriate <code>query.Aggregate</code> enum value to pass in your argument. This enum holds all the supported values for this parameter.

## Syntax



**Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

var myLocationJoin = mySalesRepJoin.autoJoin({
    fieldId: 'location'
});

var firstCondition = myCustomerQuery.createCondition({
```

```

        fieldId: 'id',
        operator: query.Operator.EQUAL,
        values: 107
    });
    var secondCondition = myCustomerQuery.createCondition({
        fieldId: 'id',
        operator: query.Operator.EQUAL,
        values: 2647
    });
    var thirdCondition = mySalesRepJoin.createCondition({
        fieldId: 'email',
        operator: query.Operator.START_WITH_NOT,
        values: 'foo'
    });

    myCustomerQuery.condition = myCustomerQuery.and(
        thirdCondition, myCustomerQuery.not(
            myCustomerQuery.or(firstCondition, secondCondition)
        )
    );
}

var resultSet = search.run();
...
// Add additional code

```

## Component.createSort(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a sort based on the <a href="#">query.Component</a> object. The <a href="#">query.Sort</a> object describes a sort that is placed on a particular query result column or condition.  To create a sort: <ul style="list-style-type: none"> <li>■ Use <a href="#">Component.createSort(options)</a> to create a sort based on a join relationship created with <a href="#">Query.autoJoin(options)</a> or <a href="#">Component.autoJoin(options)</a>.</li> <li>■ Use <a href="#">Query.createSort(options)</a> to create a sort based on the initial query definition created with <a href="#">query.create(options)</a>.</li> <li>■ Assign all created sorts as array values to <a href="#">Query.sort</a>. For an example, see <a href="#">Syntax</a>.</li> </ul>
<b>Returns</b>	<a href="#">query.Sort</a>
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Component</a>
<b>Sibling Object Members</b>	<a href="#">Component Object Members</a>
<b>Since</b>	2018.1

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.column</code>	<code>query.Column</code>	required	The query result column that you want to sort by. This value sets the <code>Sort.column</code> property.
<code>optionsascending</code>	boolean	optional	<p>Whether the sort direction is ascending. This value sets the <code>Sort.ascending</code> property.</p> <p>The default value of this property is <code>true</code>, meaning that the sort direction is ascending. If you want the sort direction to be descending, set this property to <code>false</code>.</p>
<code>options.caseSensitive</code>	boolean	optional	<p>Whether the sort is case sensitive. This value sets the <code>Sort.caseSensitive</code> property.</p> <p>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:</p> <ul style="list-style-type: none"> <li>■ Banana</li> <li>■ Orange</li> <li>■ apple</li> <li>■ grapefruit</li> <li>■ kiwi</li> </ul> <p>Here is the same list of items sorted using a regular (not case-sensitive) sort with a sort direction of ascending:</p> <ul style="list-style-type: none"> <li>■ apple</li> <li>■ Banana</li> <li>■ grapefruit</li> <li>■ kiwi</li> <li>■ Orange</li> </ul> <p>The default value of this property is <code>false</code>.</p>
<code>options.locale</code>	string	optional	<p>The locale to use for the sort. This value sets the <code>Sort.locale</code> property.</p> <p>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.</p>

Parameter	Type	Required / Optional	Description
			Use the appropriate <code>query.SortLocale</code> enum value to pass in your argument. This enum holds all the supported values for this parameter.
<code>options.nullsLast</code>	boolean	optional	<p>Whether query results with null values are listed at the end of the query results. This value sets the <code>Sort.nullsLast</code> property.</p> <p>The default value of this property is the value of the <code>options.ascending</code> property. For example, if the <code>options.ascending</code> property is set to <code>true</code>, the <code>options.nullsLast</code> property is also set to <code>true</code>.</p>

## Syntax



**Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'id'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'entityid'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'email'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'hiredate'
    })
];

myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[1]
    }),
    mySalesRepJoin.createSort({
        column: mySalesRepJoin.columns[0],
        ascending: false
    })
];
```

```

        })
];

var resultSet = myCustomerQuery.run();
...
// Add additional code

```

## Component.join(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates a join relationship. This method is an alias to <a href="#">Component.autoJoin(options)</a>.</p> <p>Use the method <a href="#">query.create(options)</a> to create your initial query definition (<a href="#">query.Query</a>). The initial query definition uses one query type. For available query types, see <a href="#">query.Type</a>.</p> <p>After you create the initial query definition, use <a href="#">Query.autoJoin(options)</a> to create your first join (<a href="#">query.Component</a>). Then use <a href="#">Component.join(options)</a> to create each subsequent join (<a href="#">query.Component</a>).</p> <p> <b>Important:</b> The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic <a href="#">Available Record Types</a>.</p>
<b>Returns</b>	<a href="#">query.Component</a>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Component</a>
<b>Sibling Object Members</b>	<a href="#">Component Object Members</a>
<b>Since</b>	2018.1

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.fieldId</code>	string	required	<p>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.</p> <p>Obtain this value from the <a href="#">Records Browser</a>:</p> <ol style="list-style-type: none"> <li>1. Go to the parent component's record type.</li> </ol>

Parameter	Type	Required / Optional	Description
			<p>2. Scroll until you see the Search Joins table.</p> <p>3. Locate the appropriate value in the Join ID column.</p> <p>For more information on the Records Browser, see <a href="#">Using the SuiteScript Records Browser</a>.</p>

## Syntax

 **Important:** The following code sample 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 myEntityJoin = myTransactionQuery.join({
    fieldId: 'entity'
});

myTransactionQuery.columns = [
    myEntityJoin.createColumn({
        fieldId: 'subsidiary'
    })
];

myTransactionQuery.sort = [
    myTransactionQuery.createSort({
        column: myTransactionQuery.columns[0],
        ascending: false
    })
];

var results = myTransactionQuery.runPaged({
    pageSize: 10
});
...
// Add additional code
```

## Component.joinFrom(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates an explicit directional join relationship from another component to this component (an inverse join). This method sets the <a href="#">Component.source</a> property on the returned <a href="#">query.Component</a> object.</p> <p>Use the method <a href="#">query.create(options)</a> to create your initial query definition (<a href="#">query.Query</a>). The initial query definition uses one query type. For available query types, see <a href="#">query.Type</a>.</p>
---------------------------	---

	After you create the initial query definition, use this method to create explicit directional joins from other components to this component.
	 <b>Important:</b> The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic <a href="#">Available Record Types</a> .
<b>Returns</b>	<code>query.Component</code>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<code>query.Component</code>
<b>Sibling Object Members</b>	Component Object Members
<b>Since</b>	2018.2

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.fieldId</code>	string	required	<p>The column type (field type) that joins the parent component to the new component.</p> <p>Obtain this value from the <a href="#">Records Browser</a>:</p> <ol style="list-style-type: none"> <li>1. Go to the parent component's record type.</li> <li>2. Scroll until you see the Search Joins table.</li> <li>3. Locate the appropriate value in the Join ID column.</li> </ol> <p>For more information on the Records Browser, see <a href="#">Using the SuiteScript Records Browser</a>.</p>
<code>options.source</code>	string	required	<p>The query type of the component joined to this component. This value sets the <a href="#">Component.source</a> property.</p> <p>This value can be described as the inverse relationship of this component, and it determines the source query type of the newly joined component.</p>

## Errors

Error Code	Thrown If
<code>RELATIONSHIP_ALREADY_USED</code>	The specified join relationship already exists.

## Syntax



**Important:** The following code sample 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 myEmployeeQuery = query.create({
    type: query.Type.EMPLOYEE
});

var mySalesOrderJoin = myEmployeeQuery.joinFrom({
    fieldId: 'salesrep',
    source: 'salesorder'
});

var items = mySalesOrderJoin.autoJoin({
    fieldId: 'item'
});

myEmployeeQuery.columns = [
    myEmployeeQuery.createColumn({
        fieldId: 'entityid'
    }),
    myEmployeeQuery.createColumn({
        fieldId: 'hiredate'
    }),
    mySalesOrderJoin.createColumn({
        fieldId: 'id'
    }),
    mySalesOrderJoin.createColumn({
        fieldId: 'trandate'
    })
];

var firstSort = myEmployeeQuery.createSort({
    column: myEmployeeQuery.columns[0],
    ascending: false
});
var secondSort = myEmployeeQuery.createSort({
    column: myEmployeeQuery.columns[1],
    ascending: true
});
myEmployeeQuery.sort = [firstSort, secondSort];

var results = myEmployeeQuery.run();
...
// Add additional code
```

## Component.joinTo(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates an explicit directional join relationship to another component from this component (a polymorphic join). This method sets the <a href="#">Component.target</a> property on the returned <a href="#">query.Component</a> object.</p> <p>Use the method <a href="#">query.create(options)</a> to create your initial query definition (<a href="#">query.Query</a>). The initial query definition uses one query type. For available query types, see <a href="#">query.Type</a>.</p> <p>After you create the initial query definition, use this method to create explicit directional joins to other components from this component.</p> <p><b>Important:</b> The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic <a href="#">Available Record Types</a>.</p>
<b>Returns</b>	<a href="#">query.Component</a>
<b>Supported Script Types</b>	<p>Client and server scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Component</a>
<b>Sibling Object Members</b>	<a href="#">Component Object Members</a>
<b>Since</b>	2018.2

## Parameters



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

Parameter	Type	Required / Optional	Description
<code>options.fieldId</code>	string	required	<p>The column type (field type) that joins the parent component to the new component.</p> <p>Obtain this value from the <a href="#">Records Browser</a>:</p> <ol style="list-style-type: none"> <li>1. Go to the parent component's record type.</li> <li>2. Scroll until you see the Search Joins table.</li> <li>3. Locate the appropriate value in the Join ID column.</li> </ol> <p>For more information on the Records Browser, see <a href="#">Using the SuiteScript Records Browser</a>.</p>
<code>options.target</code>	string	required	<p>The query type of the component joined to this component. This value sets the <a href="#">Component.target</a> property.</p> <p>This value can be described as the polymorphic relationship of this component, and it determines the target query type of the newly joined component.</p>

## Errors

Error Code	Thrown If
RELATIONSHIP_ALREADY_USED	The specified join relationship already exists.

## Syntax

 **Important:** The following code sample 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 myEntityJoin = myTransactionQuery.joinTo({
    fieldId: 'entity',
    target: query.Type.CUSTOMER
});

myTransactionQuery.columns = [
    myEntityJoin.createColumn({
        fieldId: 'subsidiary'
    })
];

myTransactionQuery.sort = [
    myTransactionQuery.createSort({
        column: myTransactionQuery.columns[0],
        ascending: false
    })
];

var results = myTransactionQuery.runPaged({
    pageSize: 10
});
...
// Add additional code
```

## Component.child

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	A reference 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 <a href="#">query.Component</a> object.  The object values are set during the execution of <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autoJoin(options)</a> . 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 Members	Component Object Members
Since	2018.1

## Syntax

 **Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

var myDeptJoin = mySalesRepJoin.autoJoin({
    fieldId: 'department'
});

var theChild = mySalesRepJoin.child;
...
// Add additional code
```

## Component.parent



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	A reference to the parent <a href="#">query.Component</a> object of this component.  This property is set during the execution of <a href="#">Query.autoJoin(options)</a> or <a href="#">Component.autoJoin(options)</a> .
<b>Type</b>	string (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Component
<b>Sibling Object Members</b>	Component Object Members
<b>Since</b>	2018.1

## Syntax

**Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

var myDeptJoin = mySalesRepJoin.autoJoin({
    fieldId: 'department'
});

var theParent = myDeptJoin.parent;
...
// Add additional code
```

## Component.source

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The query 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 <a href="#">Query.joinFrom(options)</a> and <a href="#">Component.joinFrom(options)</a> .
<b>Type</b>	string (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Component
<b>Sibling Object Members</b>	<a href="#">Component Object Members</a>
<b>Since</b>	2018.1

## Syntax

**Important:** The following code sample 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 myEmployeeQuery = query.create({
    type: query.Type.EMPLOYEE
});

var myTransactionJoin = myEmployeeQuery.joinFrom({
```

```

    fieldId: 'entity',
    source: 'transaction'
});

var theSource = myTransactionJoin.source;
...
// Add additional code

```

## Component.target



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The query 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 <a href="#">Query.joinTo(options)</a> and <a href="#">Component.joinTo(options)</a> .
<b>Type</b>	string (read-only)
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Component</a>
<b>Sibling Object Members</b>	<a href="#">Component Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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 myEmployeeJoin = myTransactionQuery.joinTo({
    fieldId: 'createdby',
    target: 'employee'
});

var theTarget = myEmployeeJoin.target;
...
// Add additional code

```

## Component.type



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The query type of this component.
-----------------------------	-----------------------------------

	This property is set during the execution of <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autoJoin(options)</a> .
<b>Type</b>	string (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.Component</a>
<b>Sibling Object Members</b>	<a href="#">Component Object Members</a>
<b>Since</b>	2018.1

## Syntax

 **Important:** The following code sample 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
});

var theType = myCustomerQuery.type;
...
// Add additional code
```

## query.Condition



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	A condition that narrows the query results. The <code>query.Condition</code> object acts in the same capacity as the <code>search.Filter</code> object in the <a href="#">N/search Module</a> . The primary difference is that <code>query.Condition</code> objects can contain other <code>query.Condition</code> objects.  To create conditions: <ul style="list-style-type: none"><li>■ Use <a href="#">Query.createCondition(options)</a> to create conditions for the initial query definition created with <code>query.create(options)</code>.</li><li>■ Use <a href="#">Component.createCondition(options)</a> to create conditions for the join relationships created with <code>Query.autoJoin(options)</code> and <code>Component.autoJoin(options)</code>.</li><li>■ If you have multiple conditions, use them to create a new nested condition with the methods <code>Query.and()</code>, <code>Query.or()</code>, and <code>Query.not()</code>.</li><li>■ Assign your simple or nested condition to <code>Query.condition</code>. For an example, see <a href="#">Syntax</a>.</li></ul>
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/query Module</a>
<b>Methods and Properties</b>	<a href="#">Condition Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

var myLocationJoin = mySalesRepJoin.autoJoin({
    fieldId: 'location'
});

var firstCondition = myCustomerQuery.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var secondCondition = myCustomerQuery.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var thirdCondition = mySalesRepJoin.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});

myCustomerQuery.condition = myCustomerQuery.and(
    thirdCondition, myCustomerQuery.not(
        myCustomerQuery.or(firstCondition, secondCondition)
    )
);

var resultSet = myCustomerQuery.run();
...
// Add additional code
```

## Condition.aggregate



**Note:** The content in this help topic pertains to SuiteScript 2.0.

### Property Description

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 <a href="#">Query.createCondition(options)</a> or <a href="#">Component.createCondition(options)</a> .
	<p><b>Note:</b> This property is not applicable to parent conditions created with the execution of <a href="#">Query.and()</a>, <a href="#">Query.or()</a>, or <a href="#">Query.not()</a>.</p>
<b>Type</b>	string (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Condition
<b>Sibling Object Members</b>	Condition Object Members
<b>Since</b>	2018.1

## Syntax

**Important:** The following code sample 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
});

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

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	An array of child conditions used to create the parent condition.
	<p><b>Note:</b> This property is applicable to only parent conditions created with the execution of <a href="#">Query.and()</a>, <a href="#">Query.or()</a>, or <a href="#">Query.not()</a>.</p>
<b>Type</b>	query.Condition[] (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Condition

<b>Sibling Object Members</b>	<a href="#">Condition Object Members</a>
<b>Since</b>	2018.1

## Syntax

 **Important:** The following code sample 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
});

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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>The component used to created the condition</p> <p>This property is set during the execution of <a href="#">Query.createCondition(options)</a> and <a href="#">Component.createCondition(options)</a>.</p> <p> <b>Note:</b> This property is not applicable to parent conditions created with the execution of <a href="#">Query.and()</a>, <a href="#">Query.or()</a>, or <a href="#">Query.not()</a>.</p>
<b>Type</b>	<a href="#">query.Component</a> (read-only)
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Condition</a>
<b>Sibling Object Members</b>	<a href="#">Condition Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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
});

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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The name of the condition.  This property is set during the execution of <a href="#">Query.createCondition(options)</a> and <a href="#">Component.createCondition(options)</a> .  <b>Note:</b> This property is not applicable to parent conditions created with the execution of <a href="#">Query.and()</a> , <a href="#">Query.or()</a> , or <a href="#">Query.not()</a> .
<b>Type</b>	string (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Condition
<b>Sibling Object Members</b>	<a href="#">Condition Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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
});

var myCondition = myCustomerQuery.createCondition({
    fieldId: 'openingbalance',
    operator: query.Operator.GREATER,
    values: 10000
});

var theFieldId = myCondition.fieldId;
...
// Add additional code

```

## Condition.formula



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The formula used to create the condition.  This property is set during the execution of <a href="#">Query.createCondition(options)</a> and <a href="#">Component.createCondition(options)</a> .  For more information on formulas, see the help topics <a href="#">Formulas in Search</a> and <a href="#">SQL Expressions</a> .
<b>Type</b>	string (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.Condition</a>
<b>Sibling Object Members</b>	<a href="#">Condition Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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 myFormulaCondition = myTransactionQuery.createCondition({
    formula: '{amount} * 125',

```

```

    operator: query.Operator.GREATER,
    values: 50000,
    type: query.ReturnType.CURRENCY
});

var theFormula = myFormulaCondition.formula;
...
// Add additional code

```

## Condition.operator



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The name of the operator used to create the condition.  This property is set during the execution of <a href="#">Query.createCondition(options)</a> and <a href="#">Component.createCondition(options)</a> .
<b>Type</b>	string (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.Condition</a>
<b>Sibling Object Members</b>	<a href="#">Condition Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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
});

var myCondition = myCustomerQuery.createCondition({
    fieldId: 'openingbalance',
    operator: query.Operator.GREATER,
    values: 10000
});

var theOperator = myCondition.operator;
...
// Add additional code

```

## Condition.type



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>The return type of the formula used to create the condition.</p> <p>This property is set during the execution of <a href="#">Query.createCondition(options)</a> or <a href="#">Component.createCondition(options)</a>.</p> <p>For more information on formulas, see the help topics <a href="#">Formulas in Search</a> and <a href="#">SQL Expressions</a>.</p> <p><b>Note:</b> This property is not applicable to parent conditions created with the execution of <a href="#">Query.and()</a>, <a href="#">Query.or()</a>, or <a href="#">Query.not()</a>.</p>
<b>Type</b>	string (read-only)
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Condition</a>
<b>Sibling Object Members</b>	<a href="#">Condition Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	An array of values used by an operator to create the condition.
-----------------------------	---

	This property is set by passing in values for <code>options.fieldId</code> , <code>options.operator</code> and <code>options.values</code> during the execution of <code>Query.createCondition(options)</code> or <code>Component.createCondition(options)</code> .
	<p> <b>Note:</b> 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>.</p>
<b>Type</b>	<code>string[]   Date[]</code> (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<code>query.Condition</code>
<b>Sibling Object Members</b>	<a href="#">Condition Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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
});

var myCondition = myCustomerQuery.createCondition({
    fieldId: 'firstname',
    operator: query.Operator.ANY_OF,
    values: ['Martin', 'Russell', 'Janina']
});

var theValues = myCondition.values;
...
// Add additional code
```

## query.Page



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	One page of the paged query results.
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/query Module</a>
<b>Methods and Properties</b>	<a href="#">Page Object Members</a>

Since	2018.1
-------	--------

## Syntax

**Important:** The following code sample 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'
    }),
    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

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The query results contained in this page.
<b>Type</b>	<code>query.ResultSet</code> (read-only)

<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Page
<b>Sibling Object Members</b>	Page Object Members
<b>Since</b>	2018.1

## Syntax

 **Important:** The following code sample 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 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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Whether the page is the first of the paged query results.
<b>Type</b>	boolean (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Page
<b>Sibling Object Members</b>	Page Object Members
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Whether the page is the last of the paged query results.
<b>Type</b>	boolean (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Page
<b>Sibling Object Members</b>	<a href="#">Page Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'firstname'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'email'
    })
];

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.pageRange

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The range of query results for this page.
<b>Type</b>	<a href="#">query.PageRange</a> (read-only)
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Page</a>
<b>Sibling Object Members</b>	<a href="#">Page Object Members</a>
<b>Since</b>	2018.1

## Syntax

 **Important:** The following code sample 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 myPagedResults = myCustomerQuery.runPaged({
    pageSize: 10
});

var iterator = myPagedResults.iterator();
iterator.each(function(resultPage) {
    var currentPage = resultPage.value;
    var thePageRange = currentPage.pageRange;
    return true;
});
...
// Add additional code

```

## Page.pagedData

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The set of paged query results that this page is from.
<b>Type</b>	<a href="#">query.PagedData</a> (read-only)
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Page</a>
<b>Sibling Object Members</b>	<a href="#">Page Object Members</a>
<b>Since</b>	2018.1

## Syntax

 **Important:** The following code sample 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 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	A set of paged query results. This object also contains information about the set of paged results it encapsulates.  Use <a href="#">Query.runPaged()</a> or <a href="#">Query.runPaged.promise()</a> to create this object.  For paged queries, the maximum number of result rows per page is 1000. The minimum number of result rows per page is 5, except for the last page in the result set (because the last page may include fewer than 5 results).
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/query Module
<b>Methods and Properties</b>	<a href="#">PagedData Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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'
    }),
    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()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Standard SuiteScript 2.0 object for iterating through results
<b>Returns</b>	Iterator object
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.PagedData</a>
<b>Sibling Object Members</b>	<a href="#">PagedData Object Members</a>
<b>Since</b>	2018.1

## Syntax

 **Important:** The following code sample 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 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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The total number of paged query result rows.
<b>Type</b>	number (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.PagedData
<b>Sibling Object Members</b>	PagedData Object Members
<b>Since</b>	2018.1

## Syntax

 **Important:** The following code sample 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 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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	An array of page ranges for the paged query results.
<b>Type</b>	<a href="#">query.PageRange[]</a>
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.PagedData</a>
<b>Sibling Object Members</b>	<a href="#">PagedData Object Members</a>
<b>Since</b>	2018.1

## Syntax

 **Important:** The following code sample 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 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The number of query result rows per page.  For paged queries, the maximum number of result rows per page is 1000. The minimum number of result rows per page is 5, except for the last page in the result set (because the last page may include fewer than 5 results).
<b>Type</b>	number (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.PagedData
<b>Sibling Object Members</b>	<a href="#">PagedData Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	The range of query results for a page.
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/query Module</a>
<b>Methods and Properties</b>	<a href="#">PageRange Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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'
    }),
    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);
}
...
// Add additional code

```

## PageRange.index



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The array index for this page range.
<b>Type</b>	number (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.PageRange</a>
<b>Sibling Object Members</b>	<a href="#">PageRange Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The number of query result rows in this page range.
<b>Type</b>	number (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.PageRange
<b>Sibling Object Members</b>	PageRange Object Members
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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 myPagedResults = myCustomerQuery.runPaged({
    pageSize: 10
});

var iterator = myPagedResults.iterator();
iterator.each(function(resultPage) {
    var currentPage = resultPage.value;
    var currentPageRange = currentPage.pageRange;
    var theSize = currentPageRange.size;
    return true;
});
...
// Add additional code

```

# query.Query



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	<p>The <code>query.Query</code> object encapsulates the query definition. To create a query with the N/query module:</p> <ol style="list-style-type: none"> <li>1. Use the <code>query.create(options)</code> method to create your query definition (this object). The initial query definition uses one query type. For available query types, see <code>query.Type</code>.</li> <li>2. After you create the initial query definition, use <code>Query.autoJoin(options)</code> to create your first join.</li> <li>3. Then use either <code>Query.autoJoin(options)</code> or <code>Component.autoJoin(options)</code> to create subsequent joins.</li> </ol> <p>The query definition always contains at least one <code>query.Component</code> object. The <code>query.Component</code> 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.</p> <p>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.</p> <p>Queries with joins contain multiple components. The query definition contains a child <code>query.Component</code> object for each of the following:</p> <ul style="list-style-type: none"> <li>■ <b>The initial query definition:</b> The initial <code>query.Component</code> object is called the root component. It encapsulates the initial query type passed to <code>query.create(options)</code>. The root component is automatically created with the initial query definition and is a child to the <code>query.Query</code> object. The <code>Query.root</code> property contains a reference to the root component.</li> <li>■ <b>The first join:</b> The second <code>query.Component</code> object is created with <code>Query.autoJoin(options)</code>. It encapsulates the relationship between the initial query definition and the second query type. This relationship is determined by the field ID passed to <code>Query.autoJoin(options)</code>. The second <code>query.Component</code> object is a child to the root component.</li> <li>■ <b>Each subsequent join:</b> The third <code>query.Component</code> object is created with <code>Query.autoJoin(options)</code> or <code>Component.autoJoin(options)</code>. All subsequent joins are also created with <code>Query.autoJoin(options)</code> or <code>Component.autoJoin(options)</code>. Each of these <code>query.Component</code> objects encapsulates the relationship between all previous query types and the new query type. This relationship is determined by the field ID passed to <code>Component.autoJoin(options)</code>.</li> </ul>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/query Module
<b>Methods and Properties</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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 myEntityJoin = myTransactionQuery.autoJoin({
    fieldId: 'entity'
});

myTransactionQuery.columns = [
    myEntityJoin.createColumn({
        fieldId: 'subsidiary'
})];

myTransactionQuery.sort = [
    myTransactionQuery.createSort({
        column: myTransactionQuery.columns[0],
        ascending: false
    })
];

var results = myTransactionQuery.runPaged({
    pageSize: 10
});
...
// Add additional code

```

## Query.and()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates a new condition (a <a href="#">query.Condition</a> object) that corresponds to a logical conjunction (AND) of the arguments passed to the method. The arguments must be one or more <a href="#">query.Condition</a> objects.</p> <p>A condition narrows the query results. The <a href="#">query.Condition</a> object acts in the same capacity as the <a href="#">search.Filter</a> object in the <a href="#">N/search Module</a>. The primary difference is that <a href="#">query.Condition</a> objects can contain other <a href="#">query.Condition</a> objects.</p> <p>To create conditions:</p> <ul style="list-style-type: none"> <li>■ Use <a href="#">Query.createCondition(options)</a> to create conditions for the initial query definition created with <a href="#">query.create(options)</a>.</li> <li>■ Use <a href="#">Component.createCondition(options)</a> to create conditions for the join relationships created with <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autoJoin(options)</a>.</li> <li>■ If you have multiple conditions, use them to create a new parent condition with the methods <a href="#">Query.and()</a>, <a href="#">Query.or()</a>, and <a href="#">Query.not()</a>.</li> <li>■ Assign your parent condition to <a href="#">Query.condition</a>. For an example, see <a href="#">Syntax</a>.</li> </ul>
<b>Returns</b>	<a href="#">query.Condition</a>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None

<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Query
<b>Sibling Object Members</b>	Query Object Members
<b>Since</b>	2018.1

## Parameters

Parameter	Type	Required / Optional	Description
condition 1 — n	query.Condition	required	<p>One or more condition objects.</p> <p>There is no limit on the number of conditions you can specify.</p>

## Syntax

 **Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

var myLocationJoin = mySalesRepJoin.autoJoin({
    fieldId: 'location'
});

var firstCondition = myCustomerQuery.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var secondCondition = myCustomerQuery.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var thirdCondition = mySalesRepJoin.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});

myCustomerQuery.condition = myCustomerQuery.and(
    thirdCondition, myCustomerQuery.not(

```

```

        myCustomerQuery.or(firstCondition, secondCondition)
    )
);

var resultSet = myCustomerQuery.run();
...
// Add additional code

```

## Query.autoJoin(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates a join relationship.</p> <p>Use the method <a href="#">query.create(options)</a> to create your initial query definition (<a href="#">query.Query</a>). The initial query definition uses one query type. For available query types, see <a href="#">query.Type</a>.</p> <p>After you create the initial query definition, use <a href="#">Query.autoJoin(options)</a> to create your first join (<a href="#">query.Component</a>). Then use <a href="#">Component.autoJoin(options)</a> to create each subsequent join (<a href="#">query.Component</a>).</p> <p> <b>Note:</b> This method is a shortcut for the chained <a href="#">Query.root</a> and <a href="#">Component.autoJoin(options)</a>: <a href="#">Query.root.join(options)</a>. The <a href="#">Query.root</a> property references the root component, which is a <a href="#">query.Component</a> object.</p> <p> <b>Important:</b> The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic <a href="#">Available Record Types</a>.</p>
<b>Returns</b>	<a href="#">query.Component</a>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.Query</a>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.2

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.fieldId</code>	string	required	The column type (field type) that joins the parent component to the new component. This value determines the columns on

Parameter	Type	Required / Optional	Description
			<p>which the components are joined and the type of the newly joined component.</p> <p>Obtain this value from the <a href="#">Records Browser</a>:</p> <ol style="list-style-type: none"> <li>1. Go to the parent component's record type.</li> <li>2. Scroll until you see the Search Joins table.</li> <li>3. Locate the appropriate value in the Join ID column.</li> </ol> <p>For more information on the Records Browser, see <a href="#">Using the SuiteScript Records Browser</a>.</p>

## Errors

Error Code	Thrown If
RELATIONSHIP_ALREADY_USED	The specified join relationship already exists.

## Syntax

**⚠ Important:** The following code sample 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 myEntityJoin = myTransactionQuery.autoJoin({
    fieldId: 'entity'
});

myTransactionQuery.columns = [
    myEntityJoin.createColumn({
        fieldId: 'subsidiary'
    })
];

myTransactionQuery.sort = [
    myTransactionQuery.createSort({
        column: myTransactionQuery.columns[0],
        ascending: false
    })
];

var results = myTransactionQuery.runPaged({
    pageSize: 10
});
...
// Add additional code
```

## Query.createColumn(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates a query result column based on the <a href="#">query.Query</a> object. The <a href="#">query.Column</a> object is the equivalent of the <a href="#">search.Column</a> object in the <a href="#">N/search Module</a>. The <a href="#">query.Column</a> object describes the field types (columns) that are displayed from the query results.</p> <p>To create columns:</p> <ul style="list-style-type: none"> <li>■ Use <a href="#">Query.createColumn(options)</a> to create columns on the initial query definition created with <a href="#">query.create(options)</a>. Use this method in one of two ways:           <ul style="list-style-type: none"> <li>□ Pass in an argument for the parameter <code>options.fieldId</code>.</li> <li>□ Pass in an argument for the parameter <code>options.formula</code>. If you use this option, you can also use the optional parameter <code>options.type</code>.</li> </ul> </li> <li>■ If needed, use <a href="#">Component.createColumn(options)</a> to create conditions on the join relationships created with <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autoJoin(options)</a>.</li> <li>■ Assign all created columns as array values to <a href="#">Query.columns</a>. For an example, see <a href="#">Syntax</a>.</li> </ul> <p>When you create a column, you can specify a field context. The field context determines how field values are displayed in the column. For example, you can specify that a column should display raw data (such as internal IDs), consolidated or converted amounts (such as currency totals), or user-friendly values (such as names). You can specify a field context in two ways:</p> <ul style="list-style-type: none"> <li>■ Use a context from the <a href="#">query.FieldContext</a> enum directly as the value of the <code>options.context</code> parameter. For example:</li> </ul> <pre>myTransactionLine.createColumn({   fieldId: 'netamount',   context: query.FieldContext.CURRENCY_CONSOLIDATED });</pre> <p>This example is the simplest way to specify a field context that does not accept additional parameters. Because the <code>options.context</code> parameter is an Object, this example is equivalent to the following:</p> <pre>myTransactionLine.createColumn({   fieldId: 'netamount',   context: {     name: query.FieldContext.CURRENCY_CONSOLIDATED   } });</pre> <ul style="list-style-type: none"> <li>■ Use a context from the <a href="#">query.FieldContext</a> enum as the value of the <code>options.context.name</code> parameter, and specify additional parameters using the <code>options.context.params</code> parameter. For example:</li> </ul> <pre>myTransactionLine.createColumn({   fieldId: 'netamount',   context: {     name: query.FieldContext.CONVERTED,     params: {       currencyId: 4,       date: new Date('2019/01/01')     }   } });</pre>
---------------------------	--

	<p>In this example, the created column displays the value of the netamount currency field using the exchange rate that was in effect on January 1, 2019 for the currency with an ID of 4.</p> <p>In this release, only the <code>query.FieldContext.CONVERTED</code> context uses additional parameters. The supported parameters are <code>currencyId</code> and <code>date</code>. For the <code>date</code> parameter, you can pass a JavaScript <code>Date</code> object or <code>query.RelativeDate</code> object. If you pass a <code>query.RelativeDate</code> object using a value from the <code>query.RelativeDateRange</code> enum, use the <code>start</code> property or <code>end</code> property to specify the exact date of the exchange rate. For example, to use the exchange rate that was in effect at the beginning of the last fiscal quarter:</p> <pre>myTransactionLine.createColumn({     fieldId: 'netamount',     context: {         name: query.FieldContext.CONVERTED,         params: {             currencyId: 4,             date: query.RelativeDateRange.LAST_FISCAL_QUARTER.start         }     } });</pre> <p>If you use only the <code>query.RelativeDate</code> object from the <code>query.RelativeDateRange</code> enum and do not specify either the <code>start</code> or <code>end</code> properties, the end date of the relative date range is used. This behavior means that the following two <code>date</code> properties are equivalent:</p> <ul style="list-style-type: none"> <li>■ <code>date: query.RelativeDateRange.LAST_FISCAL_QUARTER</code></li> <li>■ <code>date: query.RelativeDateRange.LAST_FISCAL_QUARTER.end</code></li> </ul> <p><b>Note:</b> This method is a shortcut for the chained <code>Query.root</code> and <code>Component.createColumn(options):Query.root.createColumn(options)</code>. The <code>Query.root</code> property references the root component, which is a <code>query.Component</code> object.</p>
<b>Returns</b>	<code>query.Column</code>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<code>query.Query</code>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.1

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.fieldId</code>	string	required if <code>options.formula</code> is not used	The name of the query result column. This value sets the <code>Column.fieldId</code> property.

Parameter	Type	Required / Optional	Description
			<p>Obtain this value from the <a href="#">Records Browser</a>:</p> <ol style="list-style-type: none"> <li>1. Go to the appropriate record type.</li> <li>2. Scroll until you see the Search Columns table.</li> <li>3. Locate the appropriate value in the Internal ID column.</li> </ol> <p>For more information on the Records Browser, see <a href="#">Using the SuiteScript Records Browser</a>.</p>
<code>options.formula</code>	string	required if <code>options.fieldId</code> is not used	<p>The formula used to create the query result column. This value sets the <a href="#">Column.formula</a> property.</p> <p>For more information on formulas, see the help topics <a href="#">Formulas in Search</a> and <a href="#">SQL Expressions</a>.</p>
<code>options.type</code>	string	required if <code>options.formula</code> is used	<p>If you use the <code>options.formula</code> 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 <a href="#">Column.type</a> property.</p> <p>Use the appropriate <a href="#">query.ReturnType</a> enum value to pass in your argument. This enum holds all the supported values for this parameter.</p>
<code>options.aggregate</code>	string	optional	<p>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 <a href="#">Column.aggregate</a> property.</p> <p>Use the appropriate <a href="#">query.Aggregate</a> enum value to pass in your argument. This enum holds all the supported values for this parameter.</p>
<code>options.alias</code>	string	optional	<p>The alias for the column. An alias is an alternate name for a column, and the alias is used in mapped results. This value sets the <a href="#">Column.alias</a> property.</p> <p>You must specify an alias in certain situations if you want to use <a href="#">ResultSet.asMappedResults()</a> or <a href="#">Result.asMap()</a>. For more information, see <a href="#">Column.alias</a>.</p>
<code>options.groupBy</code>	boolean	optional	<p>Indicates whether the query results are grouped by this query result column. This value sets the <a href="#">Column.groupBy</a> property.</p> <p>If you do not pass in an argument, the default value is set to <code>false</code>.</p>
<code>options.context</code>	Object	optional	<p>The field context for values in the query result column. This value sets the <a href="#">Column.context</a> property.</p> <p>If you do not pass in an argument, the default value is set to <a href="#">query.FieldContext.RAW</a>.</p>
<code>options.context.name</code>	string	required if <code>options.context</code> is used	<p>The name of the field context.</p> <p>Use the appropriate <a href="#">query.FieldContext</a> enum value to pass in your argument. This enum holds all the supported values for this parameter.</p>
<code>options.context.params</code>	Object	required if <code>options.context.name</code> has a value of <code>query</code> .	The additional parameters to use with the specified field context.

Parameter	Type	Required / Optional	Description
		FieldContext.CONVERTED	In this release, only the <code>query.FieldContext.CONVERTED</code> context uses additional parameters. The supported parameters are <code>currencyId</code> and <code>date</code> .
<code>options.context.params.currencyId</code>	number	required if <code>options.context.name</code> has a value of <code>query.FieldContext.CONVERTED</code>	The ID of the currency to convert to.
<code>options.context.params.date</code>	<code>query.RelativeDate</code>   <code>JavaScript Date</code>	required if <code>options.context.name</code> has a value of <code>query.FieldContext.CONVERTED</code>	The date to use for the actual exchange rate between the base currency and the currency to convert to.  For example, if you want to use the exchange rate that was in effect on March 3, 2019, specify a <code>query.RelativeDate</code> object or <code>JavaScript Date</code> object that represents this date.

## Syntax



**Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'id'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'entityid'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'email'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'hiredate'
    })
];

myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[1]
    })
];
```

```

mySalesRepJoin.createSort({
    column: mySalesRepJoin.columns[0],
    ascending: false
})
];

var resultSet = myCustomerQuery.run();
...
// Add additional code

```

## Query.createCondition(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates a condition (query filter) based on the <a href="#">query.Query</a> object.</p> <p>A condition narrows the query results. The <a href="#">query.Condition</a> object acts in the same capacity as the <a href="#">search.Filter</a> object in the <a href="#">N/search Module</a>. The primary difference is that <a href="#">query.Condition</a> objects can contain other <a href="#">query.Condition</a> objects.</p> <p>To create conditions:</p> <ul style="list-style-type: none"> <li>■ Use <a href="#">Query.createCondition(options)</a> to create conditions on the initial query definition created with <a href="#">query.create(options)</a>. Use this method in one of two ways:           <ul style="list-style-type: none"> <li>□ Pass in arguments for the parameters <code>options.fieldId</code>, <code>options.operator</code>, and <code>options.values</code>. The combination of these arguments translates to <code>&lt;filter column&gt;&lt;operator&gt;&lt;field value&gt;</code> (for example, 'city' equals 'Boston').</li> <li>□ Pass in an argument for the parameter <code>options.formula</code>. If you use this option, you can also use the optional parameter <code>options.type</code>.</li> </ul> </li> <li>■ If needed, use <a href="#">Component.createCondition(options)</a> to create conditions on the join relationships created with <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autoJoin(options)</a>.</li> <li>■ If you have multiple conditions, use them to create a new nested condition with the methods <a href="#">Query.and()</a>, <a href="#">Query.or()</a>, and <a href="#">Query.not()</a>.</li> <li>■ Assign your simple or nested condition to <a href="#">Query.condition</a>. For an example, see <a href="#">Syntax</a>.</li> </ul> <p><b>Note:</b> This method is a shortcut for the chained <code>Query.root</code> and <code>Component.createCondition(options)</code>: <code>Query.root.createCondition(options)</code>. The <code>Query.root</code> property references the root component, which is a <code>query.Component</code> object.</p>
<b>Returns</b>	<a href="#">query.Condition</a>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Query</a>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.1

## Parameters



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

Parameter	Type	Required / Optional	Description
<code>options.fieldId</code>	string	required if <code>options.operator</code> and <code>options.values</code> are used	<p>The name of the condition. This value sets the <a href="#">Condition.fieldId</a> property.</p> <p>Obtain this value from the <a href="#">Records Browser</a>:</p> <ol style="list-style-type: none"> <li>1. Go to the appropriate record type.</li> <li>2. Scroll until you see the Search Filters table.</li> <li>3. Locate the appropriate value in the Internal ID column.</li> </ol> <p>For more information on the Records Browser, see <a href="#">Using the SuiteScript Records Browser</a>.</p>
<code>options.operator</code>	string	required if <code>options.fieldId</code> and <code>options.values</code> are used	<p>The operator used by the condition. This value sets the <a href="#">Condition.operator</a> parameter.</p> <p>Use the appropriate <a href="#">query.Operator</a> enum value to pass in your argument. This enum holds all the supported values for this parameter.</p>
<code>options.values</code>	string[]   Date[]	required if <code>options.fieldId</code> and <code>options.operator</code> are used, and <code>options.operator</code> does <b>not</b> have a value of <code>query.Operator.EMPTY</code> or <code>query.Operator.EMPTY_NOT</code>	<p>An array of values to use for the condition. This value sets the <a href="#">Condition.values</a> property.</p>
<code>options.formula</code>	string	required if <code>options.fieldId</code> , <code>options.operator</code> , and <code>options.values</code> are <b>not</b> used	<p>The formula used to create the condition. This value sets the <a href="#">Condition.formula</a> property.</p> <p>For more information on formulas, see the help topics <a href="#">Formulas in Search</a> and <a href="#">SQL Expressions</a>.</p>
<code>options.type</code>	string	required if <code>options.formula</code> is used	<p>If you use the <code>options.formula</code> 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 <a href="#">Condition.type</a> property.</p> <p>Use the appropriate <a href="#">query.ReturnType</a> enum value to pass in your argument. This enum holds all the supported values for this parameter.</p>
<code>options.aggregate</code>	string	optional	Use this parameter to run an aggregate function on a condition. An aggregate

Parameter	Type	Required / Optional	Description
			<p>function performs a calculation on the condition values and returns a single value. This value sets the <a href="#">Condition.aggregate</a> property.</p> <p>Use the appropriate <a href="#">query.Aggregate</a> enum value to pass in your argument. This enum holds all the supported values for this parameter.</p>

## Syntax

 **Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

var myLocationJoin = mySalesRepJoin.autoJoin({
    fieldId: 'location'
});

var firstCondition = myCustomerQuery.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var secondCondition = myCustomerQuery.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var thirdCondition = mySalesRepJoin.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});

myCustomerQuery.condition = myCustomerQuery.and(
    thirdCondition, myCustomerQuery.not(
        myCustomerQuery.or(firstCondition, secondCondition)
    )
);

var resultSet = myCustomerQuery.run();
...
// Add additional code
```

## Query.createSort(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates a sort based on the <a href="#">query.Query</a> object. The <a href="#">query.Sort</a> object describes a sort that is placed on a particular query result column.</p> <p>To create a sort:</p> <ul style="list-style-type: none"> <li>■ Use <a href="#">Search.createSort(options)</a> to create a sort based on the initial query definition created with <a href="#">query.create(options)</a>.</li> <li>■ Use <a href="#">Component.createSort(options)</a> to create a sort based on a join relationship created with <a href="#">Query.autoJoin(options)</a> or <a href="#">Component.autoJoin(options)</a>.</li> <li>■ Assign all created sorts as array values to <a href="#">Query.sort</a>. For an example, see <a href="#">Syntax</a>.</li> </ul> <p><b>Note:</b> This method is a shortcut for the chained <a href="#">Query.root</a> and <a href="#">Component.createSort(options)</a>: <a href="#">Query.root.createSort(options)</a>. The <a href="#">Query.root</a> property references the root component, which is a <a href="#">query.Component</a> object.</p>
<b>Returns</b>	<a href="#">query.Sort</a>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Query</a>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.1

## Parameters



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

Parameter	Type	Required / Optional	Description
<code>options.column</code>	<a href="#">query.Column</a>	required	The query result column that you want to sort by. This value sets the <a href="#">Sort.column</a> property.
<code>optionsascending</code>	boolean	optional	<p>Whether the sort direction is ascending. This value sets the <a href="#">Sort.ascending</a> property.</p> <p>The default value of this property is <code>true</code>, meaning that the sort direction is ascending. If you want the sort direction to be descending, set this property to <code>false</code>.</p>
<code>optionscaseSensitive</code>	boolean	optional	Whether the sort is case sensitive. This value sets the <a href="#">Sort.caseSensitive</a> property.

Parameter	Type	Required / Optional	Description
			<p>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:</p> <ul style="list-style-type: none"> <li>■ Banana</li> <li>■ Orange</li> <li>■ apple</li> <li>■ grapefruit</li> <li>■ kiwi</li> </ul> <p>Here is the same list of items sorted using a regular (not case-sensitive) sort with a sort direction of ascending:</p> <ul style="list-style-type: none"> <li>■ apple</li> <li>■ Banana</li> <li>■ grapefruit</li> <li>■ kiwi</li> <li>■ Orange</li> </ul> <p>The default value of this property is <code>false</code>.</p>
<code>options.locale</code>	string	optional	<p>The locale to use for the sort. This value sets the <a href="#">Sort.locale</a> property.</p> <p>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.</p> <p>Use the appropriate <a href="#">query.SortLocale</a> enum value to pass in your argument. This enum holds all the supported values for this parameter.</p>
<code>options.nullsLast</code>	boolean	optional	<p>Whether query results with null values are listed at the end of the query results. This value sets the <a href="#">Sort.nullsLast</a> property.</p> <p>The default value of this property is the value of the <code>optionsascending</code> property. For example, if the <code>optionsascending</code> property is set to <code>true</code>, the <code>optionsnullsLast</code> property is also set to <code>true</code>.</p>

## Syntax



**Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'id'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'entityid'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'email'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'hiredate'
    })
];
myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[1]
    }),
    mySalesRepJoin.createSort({
        column: mySalesRepJoin.columns[0],
        ascending: false
    })
];
var resultSet = myCustomerQuery.run();
...
// Add additional code

```

## Query.join(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a join relationship.
	<p><b>Important:</b> This method is an alias to <a href="#">Query.autoJoin(options)</a>. Use <a href="#">Query.autoJoin(options)</a> instead of this method to create simple joins. Use <a href="#">Query.joinFrom(options)</a> and <a href="#">Query.joinTo(options)</a> to create explicit directional joins.</p>

Use the method [query.create\(options\)](#) to create your initial query definition ([query.Query](#)). The initial query definition uses one query type. For available query types, see [query.Type](#).

	<p>After you create the initial query definition, use <code>Query.join(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>).</p> <p><b>Note:</b> This method is a shortcut for the chained <code>Query.root</code> and <code>Component.join(options)</code>: <code>Query.root.join(options)</code>. The <code>Query.root</code> property references the root component, which is a <code>query.Component</code> object.</p> <p><b>Important:</b> The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic <a href="#">Available Record Types</a>.</p>
Returns	<code>query.Component</code>
Supported Script Types	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Governance	None
Module	N/query Module
Parent Object	<code>query.Query</code>
Sibling Object Members	<code>Query Object Members</code>
Since	2018.1

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.fieldId</code>	string	required	<p>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.</p> <p>Obtain this value from the <a href="#">Records Browser</a>:</p> <ol style="list-style-type: none"> <li>1. Go to the parent component's record type.</li> <li>2. Scroll until you see the Search Joins table.</li> <li>3. Locate the appropriate value in the Join ID column.</li> </ol> <p>For more information on the Records Browser, see Using the SuiteScript Records Browser.</p>

## Syntax

**Important:** The following code sample 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 myEntityJoin = myTransactionQuery.join({
    fieldId: 'entity'
});

myTransactionQuery.columns = [
    myEntityJoin.createColumn({
        fieldId: 'subsidiary'
    })
];

myTransactionQuery.sort = [
    myTransactionQuery.createSort({
        column: myTransactionQuery.columns[0],
        ascending: false
    })
];

var results = myTransactionQuery.runPaged({
    pageSize: 10
});
...
// Add additional code

```

## Query.joinFrom(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates an explicit directional join relationship from another component to this component (an inverse join). This method sets the <a href="#">Component.source</a> property on the returned <a href="#">query.Component</a> object.</p> <p>Use the method <a href="#">query.create(options)</a> to create your initial query definition (<a href="#">query.Query</a>). The initial query definition uses one query type. For available query types, see <a href="#">query.Type</a>.</p> <p>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.</p>
	<p><b>Note:</b> This method is a shortcut for the chained <a href="#">Query.root</a> and <a href="#">Component.joinFrom(options)</a>: <a href="#">Query.root.joinFrom(options)</a>. The <a href="#">Query.root</a> property references the root component, which is a <a href="#">query.Component</a> object.</p>
<b>Returns</b>	<p><a href="#">query.Component</a></p>
<b>Supported Script Types</b>	<p>Client and server scripts</p>

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Query
<b>Sibling Object Members</b>	Query Object Members
<b>Since</b>	2018.2

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.fieldId</code>	string	required	<p>The column type (field type) that joins the parent component to the new component.</p> <p>Obtain this value from the <a href="#">Records Browser</a>:</p> <ol style="list-style-type: none"> <li>1. Go to the parent component's record type.</li> <li>2. Scroll until you see the Search Joins table.</li> <li>3. Locate the appropriate value in the Join ID column.</li> </ol> <p>For more information on the Records Browser, see <a href="#">Using the SuiteScript Records Browser</a>.</p>
<code>options.source</code>	string	required	<p>The query type of the component joined to this component. This value sets the <a href="#">Component.source</a> property.</p> <p>This value can be described as the inverse relationship of this component, and it determines the source query type of the newly joined component.</p>

## Errors

Error Code	Thrown If
<code>RELATIONSHIP_ALREADY_USED</code>	The specified join relationship already exists.

## Syntax

 **Important:** The following code sample 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 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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates an explicit directional join relationship to another component from this component (a polymorphic join). This method sets the <a href="#">Component.target</a> property on the returned <a href="#">query.Component</a> object.</p> <p>Use the method <a href="#">query.create(options)</a> to create your initial query definition (<a href="#">query.Query</a>). The initial query definition uses one query type. For available query types, see <a href="#">query.Type</a>.</p> <p>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.</p> <p> <b>Note:</b> This method is a shortcut for the chained <a href="#">Query.root</a> and <a href="#">Component.joinTo(options)</a>: <code>Query.root.autoJoin(options)</code>. The <a href="#">Query.root</a> property references the root component, which is a <a href="#">query.Component</a> object.</p> <p> <b>Important:</b> The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic <a href="#">Available Record Types</a>.</p>
<b>Returns</b>	<a href="#">query.Component</a>
<b>Supported Script Types</b>	<p>Client and server scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Query</a>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>

Since	2018.2
-------	--------

## Parameters

 <b>Note:</b>	The <code>options</code> parameter is a JavaScript object.
--	--

Parameter	Type	Required / Optional	Description
<code>options.fieldId</code>	string	required	<p>The column type (field type) that joins the parent component to the new component.</p> <p>Obtain this value from the <a href="#">Records Browser</a>:</p> <ol style="list-style-type: none"> <li>1. Go to the parent component's record type.</li> <li>2. Scroll until you see the Search Joins table.</li> <li>3. Locate the appropriate value in the Join ID column.</li> </ol> <p>For more information on the Records Browser, see <a href="#">Using the SuiteScript Records Browser</a>.</p>
<code>options.target</code>	string	required	<p>The query type of the component joined to this component. This value sets the <a href="#">Component.target</a> property.</p> <p>This value can be described as the polymorphic relationship of this component, and it determines the target query type of the newly joined component.</p>

## Errors

Error Code	Thrown If
<code>RELATIONSHIP_ALREADY_USED</code>	The specified join relationship already exists.

## Syntax

 <b>Important:</b>	The following code sample shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/query Module Script Samples</a> .
---	--

```
// 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()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Executes the query and returns the query result set.  This method returns a maximum of 5000 results in the query result set. If a query matches more than 5000 results, you must use <a href="#">Query.runPaged()</a> or <a href="#">Query.runPaged.promise()</a> to retrieve the full set of results.
<b>Returns</b>	<a href="#">query.ResultSet</a>
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Query</a>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

myCustomerQuery.columns = [

```

```

myCustomerQuery.createColumn({
    fieldId: 'entityid'
}),
myCustomerQuery.createColumn({
    fieldId: 'id'
}),
mySalesRepJoin.createColumn({
    fieldId: 'entityid'
}),
mySalesRepJoin.createColumn({
    fieldId: 'email'
}),
mySalesRepJoin.createColumn({
    fieldId: 'hiredate'
})
);

myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[1]
    }),
    mySalesRepJoin.createSort({
        column: mySalesRepJoin.columns[0],
        ascending: false
    })
];
;

var resultSet = myCustomerQuery.run();
...
// Add additional code

```

## Query.run.promise()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Executes the query asynchronously and returns the query result set.
	<p><b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">Query.run()</a>. For more information on promises, see <a href="#">Promise Object</a>.</p>
<b>Returns</b>	Promise Object
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Query</a>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.1

## Query.runPaged()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Executes the query and returns a set of paged results.  For paged queries, the maximum number of result rows per page is 1000. The minimum number of result rows per page is 5, except for the last page in the result set (because the last page may include fewer than 5 results).
<b>Returns</b>	<a href="#">query.PagedData</a>
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Query</a>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.1

## Parameters

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

Parameter	Type	Required / Optional	Description
options.pageSize	string	optional	The size of each page in the query results.  The default page size is 50 results per page.

## Syntax

 **Important:** The following code sample 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 myEntityJoin = myTransactionQuery.autoJoin({
    fieldId: 'entity'
});

myTransactionQuery.columns = [
    myEntityJoin.createColumn({
        name: 'subsidiary'
    })
];
```

```

myTransactionQuery.sort = [
    myTransactionQuery.createSort({
        column: myTransactionQuery.columns[0],
        ascending: false
    })
];

var results = myTransactionQuery.runPaged({
    pageSize: 10
});

// Use the count property to count the
// search results easily
var resultCount = myTransactionQuery.runPaged({
    pageSize: 10
}).count;
...
// Add additional code

```

## Query.runPaged.promise()

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Executes the query asynchronously and returns a set of paged results.
	<p><b>i Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">Query.runPaged()</a>. For more information on promises, see <a href="#">Promise Object</a>.</p>
<b>Returns</b>	Promise Object
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Query</a>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.1

## Query.not()

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a new condition (a <a href="#">query.Condition</a> object) that corresponds to a logical negation (NOT) of the argument passed to the method. The argument must be a <a href="#">query.Condition</a> object.
---------------------------	--

	<p>A condition narrows the query results. The <a href="#">query.Condition</a> object acts in the same capacity as the <a href="#">search.Filter</a> object in the <a href="#">N/search Module</a>. The primary difference is that <a href="#">query.Condition</a> objects can contain other <a href="#">query.Condition</a> objects.</p> <p>To create conditions:</p> <ul style="list-style-type: none"> <li>▪ Use <a href="#">Query.createCondition(options)</a> to create conditions for the initial query definition created with <a href="#">query.create(options)</a>.</li> <li>▪ Use <a href="#">Component.createCondition(options)</a> to create conditions for the join relationships created with <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autoJoin(options)</a>.</li> <li>▪ If you have multiple conditions, use them to create a new parent condition with the methods <a href="#">Query.and()</a>, <a href="#">Query.or()</a>, and <a href="#">Query.not()</a>.</li> <li>▪ Assign your parent condition to <a href="#">Query.condition</a>. For an example, see <a href="#">Syntax</a>.</li> </ul>
<b>Returns</b>	<a href="#">query.Condition</a>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.Query</a>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.1

## Parameters

Parameter	Type	Required / Optional	Description
condition	<a href="#">query.Condition</a>	required	One condition object.

## Syntax

**Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

var myLocationJoin = mySalesRepJoin.autoJoin({
    fieldId: 'location'
});

var firstCondition = myCustomerQuery.createCondition({
```

```

        fieldId: 'id',
        operator: query.Operator.EQUAL,
        values: 107
    });
    var secondCondition = myCustomerQuery.createCondition({
        fieldId: 'id',
        operator: query.Operator.EQUAL,
        values: 2647
    });
    var thirdCondition = mySalesRepJoin.createCondition({
        fieldId: 'email',
        operator: query.Operator.START_WITH_NOT,
        values: 'foo'
    });

    myCustomerQuery.condition = myCustomerQuery.and(
        thirdCondition, myCustomerQuery.not(
            myCustomerQuery.or(firstCondition, secondCondition)
        )
    );
}

var resultSet = myCustomerQuery.run();
...
// Add additional code

```

## Query.or()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates a new condition (a <a href="#">query.Condition</a> object) that corresponds to a logical disjunction (OR) of the arguments passed to the method. The arguments must be one or more <a href="#">query.Condition</a> objects.</p>
	<p>A condition narrows the query results. The <a href="#">query.Condition</a> object acts in the same capacity as the <a href="#">search.Filter</a> object in the <a href="#">N/search Module</a>. The primary difference is that <a href="#">query.Condition</a> objects can contain other <a href="#">query.Condition</a> objects.</p>
	<p>To create conditions:</p>
	<ul style="list-style-type: none"> <li>■ Use <a href="#">Query.createCondition(options)</a> to create conditions for the initial query definition created with <a href="#">query.create(options)</a>.</li> <li>■ Use <a href="#">Component.createCondition(options)</a> to create conditions for the join relationships created with <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autoJoin(options)</a>.</li> <li>■ If you have multiple conditions, use them to create a new parent condition with the methods <a href="#">Query.and()</a>, <a href="#">Query.or()</a>, and <a href="#">Query.not()</a>.</li> <li>■ Assign your parent condition to <a href="#">Query.condition</a>. For an example, see <a href="#">Syntax</a>.</li> </ul>
<b>Returns</b>	<a href="#">query.Condition</a>
<b>Supported Script Types</b>	Client and server scripts
	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/query Module</a>

<b>Parent Object</b>	query.Query
<b>Sibling Object Members</b>	Query Object Members
<b>Since</b>	2018.1

## Parameters

Parameter	Type	Required / Optional	Description
condition 1 — n	query.Condition	required	<p>One or more condition objects.</p> <p>There is no limit on the number of conditions you can specify.</p>

## Syntax

**⚠ Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

var myLocationJoin = mySalesRepJoin.autoJoin({
    fieldId: 'location'
});

var firstCondition = myCustomerQuery.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var secondCondition = myCustomerQuery.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var thirdCondition = mySalesRepJoin.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'});

myCustomerQuery.condition = myCustomerQuery.and(
    thirdCondition, myCustomerQuery.not(
        myCustomerQuery.or(firstCondition, secondCondition)
    )
);
```

```
var resultSet = myCustomerQuery.run();
...
// Add additional code
```

## Query.toSuiteQL()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Converts this <a href="#">query.Query</a> object to its corresponding SuiteQL representation.</p> <p>This method returns a <a href="#">query.SuiteQL</a> object that represents the same query as the original <a href="#">query.Query</a> object. This object includes the <a href="#">SuiteQL.columns</a>, <a href="#">SuiteQL.params</a>, <a href="#">SuiteQL.query</a>, and <a href="#">SuiteQL.type</a> properties. You can run the query using <a href="#">SuiteQL.run()</a>, or you can run the query as a paged query using <a href="#">SuiteQL.runPaged(options)</a>.</p>
<b>Important:</b>	The resulting SuiteQL query string (contained in the <a href="#">SuiteQL.query</a> property) does not include any aliases you set on query result columns in the original <a href="#">query.Query</a> object. For more information about aliases, see <a href="#">Column.alias</a> .
<b>Returns</b>	<a href="#">query.SuiteQL</a>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.Query</a>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2020.1

## Syntax



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

TBD

## Query.child



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	A reference 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 <a href="#">query.Component</a> object.
-----------------------------	--

	The object values are set with the execution of <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autoJoin(options)</a> . The order of the key/value pairs reflects the parent/child hierarchy.
<b>Type</b>	Object
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.Query</a>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.1

## Syntax

 **Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

var myTaskJoin = myCustomerQuery.autoJoin({
    fieldId: 'task'
});

var theChild = myCustomerQuery.child;
...
// Add additional code
```

## Query.columns

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	An array of result columns ( <a href="#">query.Column</a> objects) returned from the query.  The <a href="#">query.Column</a> object is the equivalent of the <a href="#">search.Column</a> object in the <a href="#">N/search Module</a> . The <a href="#">query.Column</a> object describes a field type (column) that is returned from the query results.  To create columns: <ul style="list-style-type: none"><li>■ Use <a href="#">Query.createColumn(options)</a> to create conditions on the initial query definition created with <a href="#">query.create(options)</a>.</li><li>■ Use <a href="#">Component.createColumn(options)</a> to create conditions on the join relationships created with <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autoJoin(options)</a>.</li><li>■ Assign all created columns as array values to <a href="#">Query.columns</a>. For an example, see <a href="#">Syntax</a>.</li></ul>
-----------------------------	---

Type	<code>query.Column[]</code>
Module	N/query Module
Parent Object	<code>query.Query</code>
Sibling Object Members	<a href="#">Query Object Members</a>
Since	2018.1

## Syntax

 **Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'firstname'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'email'
    })
];
...
// Add additional code
```

## Query.condition

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>The simple or nested condition (a <code>query.Condition</code> object) that narrows the query results.</p> <p>The <code>query.Condition</code> object acts in the same capacity as the <code>search.Filter</code> object in the <a href="#">N/search Module</a>. The primary difference is that <code>query.Condition</code> objects can contain other <code>query.Condition</code> objects.</p> <p>To create conditions:</p> <ul style="list-style-type: none"> <li>■ Use <code>Query.createCondition(options)</code> to create conditions for the initial query definition created with <code>query.create(options)</code>.</li> </ul>
-----------------------------	---

	<ul style="list-style-type: none"> <li>■ Use <a href="#">Component.createCondition(options)</a> to create conditions for the join relationships created with <a href="#">Query.autoJoin(options)</a> and <a href="#">Component.autoJoin(options)</a>.</li> <li>■ If you have multiple conditions, use them to create a new nested condition with the methods <a href="#">Query.and()</a>, <a href="#">Query.or()</a>, and <a href="#">Query.not()</a>.</li> <li>■ Assign your simple or nested condition to <a href="#">Query.condition</a>. For an example, see <a href="#">Syntax</a>.</li> </ul>
Type	<a href="#">query.Condition</a>
Module	N/query Module
Parent Object	<a href="#">query.Query</a>
Sibling Object Members	<a href="#">Query Object Members</a>
Since	2018.1

## Syntax

 **Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>The ID of the query definition.</p> <p>This property has a value only for existing queries that are loaded using <code>query.load(options)</code>. If you create a query using <code>query.create(options)</code> but do not save it, this property is null.</p>
<b>Type</b>	number (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.Query</a>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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 myLoadedQuery = query.load({
    id: 'custworkbook237'
});

var theId = myLoadedQuery.id;
...
// Add additional code
```

## Query.name



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The name of the query definition.
-----------------------------	-----------------------------------

	This property has a value only for existing queries that are loaded using <code>query.load(options)</code> . If you create a query using <code>query.create(options)</code> but do not save it, this property is null.
	<p><b>Important:</b> The N/query module lets you create and run queries using the SuiteAnalytics Workbook query engine. 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 <a href="#">Navigating SuiteAnalytics Workbook</a>.</p>
<b>Type</b>	string (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<code>query.Query</code>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.1

## Syntax

**Important:** The following code sample 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 myLoadedQuery = query.load({
    id: 'custworkbook237'
});

var theName = myLoadedQuery.name;
...
// Add additional code
```

## Query.root

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The root component of the query definition.  The initial <code>query.Component</code> object is called the root component. It encapsulates the initial query type passed to <code>query.create(options)</code> . The root component is automatically created with the <code>query.Query</code> object and is a child of the <code>query.Query</code> object.
<b>Type</b>	<code>query.Component</code> (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<code>query.Query</code>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.1

## Syntax

**Important:** The following code sample 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
});

var theRoot = myCustomerQuery.root;
...
// Add additional code
```

## Query.sort

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	An array of query result columns ( <a href="#">query.Column</a> objects) used for sorting. This object encapsulates a sort based on the <a href="#">query.Query</a> or <a href="#">query.Component</a> object. The <a href="#">query.Sort</a> object describes a sort that is placed on a particular query result column.  To create a sort: <ul style="list-style-type: none"> <li>■ Use <a href="#">Query.createSort(options)</a> to create a sort based on the initial query definition created with <a href="#">query.create(options)</a>.</li> <li>■ Use <a href="#">Component.createSort(options)</a> to create a sort based on a join relationship created with <a href="#">Query.autoJoin(options)</a> or <a href="#">Component.autoJoin(options)</a>.</li> <li>■ Assign all created sorts as array values to <a href="#">Query.sort</a>. For an example, see <a href="#">Syntax</a>.</li> </ul>
<b>Type</b>	<a href="#">query.Sort[]</a>
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Query</a>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.1

## Syntax

**Important:** The following code sample 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
});
```

```

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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The initial query type of the query definition. This property is set during the execution of <a href="#">query.create(options)</a> .
<b>Type</b>	string (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.Query</a>
<b>Sibling Object Members</b>	<a href="#">Query Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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
});

var theType = myCustomerQuery.type;
...

// Add additional code

```

## query.RelativeDate



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	A relative date to use in query conditions.  Use <a href="#">query.createRelativeDate(options)</a> to create this object. After you create this object, you can use it in the <b>values</b> parameter of <a href="#">Query.createCondition(options)</a> or <a href="#">Component.createCondition(options)</a> .  This object represents a specific moment in time, and you can use it to create query conditions using operators from the <a href="#">query.Operator</a> enum, such as <code>query.Operator.AFTER</code> , <code>query.Operator.BEFORE</code> , and <code>query.Operator.WITHIN</code> . For more information about relative dates, see <a href="#">Relative Dates in the N/query Module</a> .
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/query Module</a>
<b>Methods and Properties</b>	<a href="#">RelativeDate Object Members</a>
<b>Since</b>	2019.1

## Syntax



**Important:** The following code sample 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 myEndDate = query.createRelativeDate({
    dateId: query.DateId.WEEKS_AGO,
    value: 2
});

var myComplexCondition = myQuery.createCondition({
    fieldId: 'trandate',
    operator: query.Operator.WITHIN,
    values: [query.RelativeDateRange.THREE_FISCAL_YEARS_AGO.start, myEndDate]
});
...

```

```
// Add additional code
```

## RelativeDate.dateId



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The date ID of the relative date.  For relative dates that you create using <code>query.createRelativeDate(options)</code> , the value of this property is set when that method is executed. For relative dates that are included in the <code>query.RelativeDateRange</code> enum, the value of this property is always available (for example, <code>query.RelativeDateRange.YESTERDAY.dateId</code> ).  This property uses values from the <code>query.DateId</code> enum.
<b>Type</b>	string (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.RelativeDate</a>
<b>Sibling Object Members</b>	<a href="#">RelativeDate Object Members</a>
<b>Since</b>	2019.1

## Syntax



**Important:** The following code sample 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 myRelativeDate = query.createRelativeDate({
    dateId: query.DateId.WEEKS_AGO,
    value: 2
});

var theDateId = myRelativeDate.dateId;
...
// Add additional code
```

## RelativeDate.end



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The end of the relative date.  For relative dates that you create using <code>query.createRelativeDate(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> ).
<b>Type</b>	Object (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<code>query.RelativeDate</code>
<b>Sibling Object Members</b>	<code>RelativeDate Object Members</code>
<b>Since</b>	2019.1

## Syntax

 **Important:** The following code sample 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 myRelativeDate = query.createRelativeDate({
    dateId: query.DateId.WEEKS_AGO,
    value: 2
});

var theEnd = myRelativeDate.end;
...
// Add additional code
```

## RelativeDate.interval

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The interval that the relative date represents.  For relative dates that you create using <code>query.createRelativeDate(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.interval</code> ).
<b>Type</b>	Object (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<code>query.RelativeDate</code>
<b>Sibling Object Members</b>	<code>RelativeDate Object Members</code>
<b>Since</b>	2019.1

## Syntax

**Important:** The following code sample 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 myRelativeDate = query.createRelativeDate({
    dateId: query.DateId.WEEKS_AGO,
    value: 2
});

var theInterval = myRelativeDate.interval;
...
// Add additional code
```

## RelativeDate.isRange

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Whether the relative date represents a range of dates or a specific moment in time. For relative date ranges that you obtain from the <a href="#">query.RelativeDateRange</a> enum, the value of this property is <b>true</b> (the relative date represents a range of dates). For all other relative dates (such as those that you create using <a href="#">query.createRelativeDate(options)</a> ), the value of this property is <b>false</b> (the relative date represents a specific moment in time).
<b>Type</b>	boolean (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.RelativeDate</a>
<b>Sibling Object Members</b>	<a href="#">RelativeDate Object Members</a>
<b>Since</b>	2019.1

## Syntax

**Important:** The following code sample 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 myRelativeDate = query.createRelativeDate({
    dateId: query.DateId.WEEKS_AGO,
    value: 2
});

// isARange is false
var isARange = myRelativeDate.isRange;
```

```
// isAnotherRange is true
var isAnotherRange = query.RelativeDateRange.LAST_MONTH_ONE_FISCAL_YEAR_AGO.isRange;
...
// Add additional code
```

## RelativeDate.start

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The start of the relative date. For relative dates that you create using <a href="#">query.createRelativeDate(options)</a> , the value of this property is set when that method is executed. For relative date ranges that are included in the <a href="#">query.RelativeDateRange</a> enum, the value of this property is always available (for example, <a href="#">query.RelativeDateRange.YESTERDAY.start</a> ).
<b>Type</b>	Object (read-only)
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.RelativeDate</a>
<b>Sibling Object Members</b>	<a href="#">RelativeDate Object Members</a>
<b>Since</b>	2019.1

## Syntax

 **Important:** The following code sample 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 myRelativeDate = query.createRelativeDate({
    dateId: query.DateId.WEEKS_AGO,
    value: 2
});

var theStart = myRelativeDate.start;
...
// Add additional code
```

## RelativeDate.value

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The value of the relative date range. For relative dates that you create using <a href="#">query.createRelativeDate(options)</a> , 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 undefined (for example, <code>query.RelativeDateRange.YESTERDAY.value</code> is undefined).
<b>Type</b>	number (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<code>query.RelativeDate</code>
<b>Sibling Object Members</b>	<a href="#">RelativeDate Object Members</a>
<b>Since</b>	2019.1

## Syntax

 **Important:** The following code sample 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 myRelativeDate = query.createRelativeDate({
    dateId: query.DateId.WEEKS_AGO,
    value: 2
});

var theValue = myRelativeDate.value;
...
// Add additional code
```

## query.Result

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	A single row of the result set ( <code>query.ResultSet</code> ).
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/query Module
<b>Methods and Properties</b>	<a href="#">Result Object Members</a>
<b>Since</b>	2018.1

## Syntax

 **Important:** The following code sample 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'
    }),
    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.asMap()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the query result as a mapped result. A mapped result is a JavaScript object with key-value pairs. In this object, the key is either the field ID or the alias that was used for the corresponding <a href="#">query.Column</a> object.
<b>Returns</b>	Object
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Result</a>
<b>Sibling Object Members</b>	<a href="#">Result Object Members</a>

Since	2019.2
-------	--------

## Syntax

**⚠ Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid',
        alias: 'cust'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'id'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'entityid'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'email'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'hiredate'
    })
];

var resultSet = myCustomerQuery.run();

for (var i = 0; i < resultSet.results.length; i++) {
    var mResult = resultSet.results[i].asMap();
    log.debug(mResult);
}
...
// Add additional code
```

## Result.values



**Note:** The content in this help topic pertains to SuiteScript 2.0.

### Property Description

The result values. Value types correspond to the [ResultSet.types](#) property. Array values correspond to the array values for [ResultSet.columns](#).

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 sample 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'
    }),
    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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	<p>The set of results returned by the query. Use <a href="#">Query.run()</a> or <a href="#">Query.run.promise()</a> to create this object.</p> <p>The maximum number of results in a <b>ResultSet</b> object is 5000. If a query matches more than 5000 results, you must use <a href="#">Query.runPaged()</a> or <a href="#">Query.runPaged.promise()</a> to retrieve the full set of results.</p>
<b>Supported Script Types</b>	<p>Client and server scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Module</b>	<a href="#">N/query Module</a>

<b>Methods and Properties</b>	ResultSet Object Members
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'email'
    })
];

var resultSet = myCustomerQuery.run();

var results = resultSet.results;
for (var i = results.length - 1; i >= 0; i--)
    log.debug(results[i].values);
...
// Add additional code
```

## ResultSet.asMappedResults()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the query result set as an array of mapped results. A mapped result is a JavaScript object with key-value pairs. In this object, the key is either the field ID or the alias that was used for the corresponding <a href="#">query.Column</a> object. When you call this method, <a href="#">Result.asMap()</a> is called on each <a href="#">query.Result</a> object in the result set.
<b>Returns</b>	Object[]
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.ResultSet</a>

<b>Sibling Object Members</b>	ResultSet Object Members
<b>Since</b>	2019.2

## Syntax

**⚠ Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid',
        alias: 'cust'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'id'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'entityid'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'email'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'hiredate'
    })
];

var resultSet = myCustomerQuery.run();
var mrSet = resultSet.asMappedResults();
...
// Add additional code
```

## ResultSet.iterator()

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Standard SuiteScript 2.0 object for iterating through results
<b>Returns</b>	Iterator object

<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.ResultSet</a>
<b>Sibling Object Members</b>	<a href="#">ResultSet Object Members</a>
<b>Since</b>	2018.1

## Syntax

**⚠ Important:** The following code sample 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'
    }),
    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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	An array of query return column references. The <code>ResultSet.columns</code> array values correspond with the <code>ResultSet.types</code> array values.
<b>Type</b>	<code>query.Column[]</code> (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<code>query.ResultSet</code>
<b>Sibling Object Members</b>	<code>ResultSet</code> Object Members
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'email'
    })
];

var queryResultSet = myCustomerQuery.run();

var theColumns = queryResultSet.columns;
...
// Add additional code
```

## ResultSet.results



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	An array of <code>query.Result</code> objects.
<b>Type</b>	<code>query.Result[]</code> (read-only)
<b>Module</b>	N/query Module
<b>Parent Object</b>	<code>query.ResultSet</code>

<b>Sibling Object Members</b>	<a href="#">ResultSet Object Members</a>
<b>Since</b>	2018.1

## Syntax

**Important:** The following code sample 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'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'email'
    })
];

var queryResultSet = myCustomerQuery.run();

var theResults = queryResultSet.results;
...
// Add additional code
```

## ResultSet.types

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	An array of the return types for <a href="#">ResultSet.results</a> . The <b>ResultSet.types</b> array values correspond with the <a href="#">ResultSet.columns</a> array values.
<b>Type</b>	string[] (read-only)
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.ResultSet</a>
<b>Sibling Object Members</b>	<a href="#">ResultSet Object Members</a>
<b>Since</b>	2018.1

## Syntax

**Important:** The following code sample 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'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'email'
    })
];

var queryResultSet = myCustomerQuery.run();

var theTypes = queryResultSet.types;
...

// Add additional code

```

## query.Sort



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	A sort based on the <a href="#">query.Query</a> or <a href="#">query.Component</a> object. The <a href="#">query.Sort</a> object describes a sort that is placed on a particular query result column.  To create a sort: <ul style="list-style-type: none"> <li>■ Use <a href="#">Query.createSort(options)</a> to create a sort based on the initial query definition created with <a href="#">query.create(options)</a>.</li> <li>■ Use <a href="#">Component.createSort(options)</a> to create a sort based on a join relationship created with <a href="#">Query.autoJoin(options)</a> or <a href="#">Component.autoJoin(options)</a>.</li> <li>■ Assign all created sorts as array values to <a href="#">Query.sort</a>. For an example, see <a href="#">Syntax</a>.</li> </ul>
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/query Module
<b>Methods and Properties</b>	<a href="#">Sort Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'id'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'entityid'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'email'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'hiredate'
    })
];
myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[1]
    }),
    mySalesRepJoin.createSort({
        column: mySalesRepJoin.columns[0],
        ascending: false
    })
];
var resultSet = myCustomerQuery.run();
...
// Add additional code

```

## Sort.ascending



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>Whether the sort direction is ascending.</p> <p>This property is set during the execution of <a href="#">Query.createSort(options)</a> and <a href="#">Component.createSort(options)</a>.</p> <p>The default value of this property is <b>true</b>, meaning that the sort direction is ascending. If you want the sort direction to be descending, set this property to <b>false</b>.</p>
<b>Type</b>	boolean

<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Sort
<b>Sibling Object Members</b>	Sort Object Members
<b>Since</b>	2018.2

## Syntax

**⚠ Important:** The following code sample 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'
    })
];

myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[0],
        ascending: false,
        caseSensitive: true,
        locale: query.SortLocale.EN_CA,
        nullsLast: false
    })
];
var theAscending = myCustomerQuery.sort[0].ascending;
...
// Add additional code
```

## Sort.caseSensitive

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Whether the sort is case sensitive.  This property is set during the execution of <a href="#">Query.createSort(options)</a> and <a href="#">Component.createSort(options)</a> .  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
-----------------------------	---

	<p>example, the following list of items is sorted using a case-sensitive sort with a sort direction of ascending:</p> <ul style="list-style-type: none"> <li>■ Banana</li> <li>■ Orange</li> <li>■ apple</li> <li>■ grapefruit</li> <li>■ kiwi</li> </ul> <p>Here is the same list of items sorted using a regular (not case-sensitive) sort with a sort direction of ascending:</p> <ul style="list-style-type: none"> <li>■ apple</li> <li>■ Banana</li> <li>■ grapefruit</li> <li>■ kiwi</li> <li>■ Orange</li> </ul> <p>The default value of this property is <b>false</b>.</p>
<b>Type</b>	boolean
<b>Module</b>	N/query Module
<b>Parent Object</b>	query.Sort
<b>Sibling Object Members</b>	<a href="#">Sort Object Members</a>
<b>Since</b>	2018.2

## Syntax



**Important:** The following code sample 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'
    })
];

myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[0],
        ascending: false,
        caseSensitive: true,
        locale: query.SortLocale.EN_CA,
        nullsLast: false
    })
]
```

```

];
var theCaseSensitive = myCustomerQuery.sort[0].caseSensitive;
...
// Add additional code

```

## Sort.column



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The query result column that the query results are sorted by. This property is set during the execution of <a href="#">Query.createSort(options)</a> and <a href="#">Component.createSort(options)</a> .
<b>Type</b>	<a href="#">query.Column</a> (read-only)
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.Sort</a>
<b>Sibling Object Members</b>	<a href="#">Sort Object Members</a>
<b>Since</b>	2018.1

## Syntax



**Important:** The following code sample 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'
    })
];

myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[0],
        ascending: false,
        caseSensitive: true,
        locale: query.SortLocale.EN_CA,
        nullsLast: false
    })
];

var theColumn = myCustomerQuery.sort[0].column;
...

```

```
// Add additional code
```

## Sort.locale



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The locale to use for the sort.  This property uses values from the <a href="#">query.SortLocale</a> enum. This property is set during the execution of <a href="#">Query.createSort(options)</a> and <a href="#">Component.createSort(options)</a> .  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.
<b>Type</b>	string
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.Sort</a>
<b>Sibling Object Members</b>	<a href="#">Sort Object Members</a>
<b>Since</b>	2018.2

## Syntax



**Important:** The following code sample 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'
    })
];

myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[0],
        ascending: false,
        caseSensitive: true,
        locale: query.SortLocale.EN_CA,
        nullsLast: false
    })
];
```

```
var theLocale = myCustomerQuery.sort[0].locale;
...
// Add additional code
```

## Sort.nullsLast



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Whether query results with null values are listed at the end of the query results.  This property is set during the execution of <a href="#">Query.createSort(options)</a> and <a href="#">Component.createSort(options)</a> .  The default value of this property is the value of the <a href="#">Sort ascending</a> property. For example, if the <a href="#">Sort ascending</a> property is set to <code>true</code> , the <a href="#">Sort nullsLast</a> property is also set to <code>true</code> .
<b>Type</b>	boolean
<b>Module</b>	N/query Module
<b>Parent Object</b>	<a href="#">query.Sort</a>
<b>Sibling Object Members</b>	<a href="#">Sort Object Members</a>
<b>Since</b>	2018.2

## Syntax



**Important:** The following code sample 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'
    })
];

myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[0],
        ascending: false,
        caseSensitive: true,
        locale: query.SortLocale.EN_CA,
        nullsLast: false
    })
];
```

```
var theNullsLast = myCustomerQuery.sort[0].nullsLast;
...
// Add additional code
```

## query.SuiteQL



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	A SuiteQL query.  SuiteQL is a query language based on the SQL-92 revision of the SQL database query language. It provides advanced query capabilities you can use to access your NetSuite records and data.  Use <a href="#">Query.toSuiteQL()</a> to create this object. This method converts an existing <a href="#">query.Query</a> object to its corresponding SuiteQL representation as a <a href="#">query.SuiteQL</a> object. You can use <a href="#">SuiteQL.run()</a> to run the query and obtain the results as a <a href="#">query.ResultSet</a> object. You can also use <a href="#">SuiteQL.runPaged(options)</a> to run the query as a paged query and obtain the results as a <a href="#">query.PagedData</a> object.  When you convert a <a href="#">query.Query</a> object to a <a href="#">query.SuiteQL</a> object, the resulting SuiteQL query is the same as the original query. It includes the same query result columns, sort order, and conditions that were set on the original query. When you run the resulting SuiteQL query using <a href="#">SuiteQL.run()</a> or <a href="#">SuiteQL.runPaged(options)</a> , you receive the same results as you would if you ran the original query using <a href="#">Query.run()</a> or <a href="#">Query.runPaged()</a> .  For more information and examples of using SuiteQL in the N/query module, see <a href="#">SuiteQL in the N/query Module</a> . For more information about SuiteQL in general, see the help topic <a href="#">SuiteQL</a> .
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/query Module</a>
<b>Methods and Properties</b>	<a href="#">SuiteQL Object Members</a>
<b>Since</b>	2020.1

## Syntax



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

TBD

## SuiteQL.columns



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The result columns to be returned from the query.
-----------------------------	---

	<p>This property is an array of <code>query.Column</code> objects. When you use <code>Query.toSuiteQL()</code> to convert an existing <code>query.Query</code> object to SuiteQL, this property contains the same <code>query.Column</code> objects that were specified for the original query.</p> <p><b>Important:</b> The SuiteQL query string in a <code>query.SuiteQL</code> object (contained in the <code>SuiteQL.query</code> property) does not include any aliases you set on query result columns in the original <code>query.Query</code> object. For more information about aliases, see <a href="#">Column.alias</a>.</p>
<b>Type</b>	<code>query.Column[]</code>
<b>Module</b>	N/query Module
<b>Parent Object</b>	<code>query.SuiteQL</code>
<b>Sibling Object Members</b>	<a href="#">SuiteQL Object Members</a>
<b>Since</b>	2020.1

## Syntax

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

TBD

## SuiteQL.params

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>The parameters for the query.</p> <p>In SuiteQL, query conditions are represented using the <code>WHERE</code> clause and a set of parameters. In a SuiteQL string, parameter values for conditions are represented using question marks (?), and the <code>params</code> property includes a list of the parameter values to use when the query runs. If the query uses more than one parameter, the order of the values in the <code>params</code> property matches the order the parameters appear in the query string.</p> <p>For example, consider the following <code>query.Condition</code> object in a query for customer records:</p> <pre>myQueryObject.createCondition({     fieldId: 'creditlimit',     operator: query.Operator.LESS_OR_EQUAL,     values: [50000] });</pre> <p>If you use <code>Query.toSuiteQL()</code> to convert this query to SuiteQL, the resulting SuiteQL query string includes the following <code>WHERE</code> clause:</p> <pre>WHERE customer.creditlimit &lt;= ?</pre> <p>In the resulting <code>query.SuiteQL</code> object, the <code>params</code> property includes the value <code>50000</code>.</p> <p>This property is read-only. After you create a <code>query.SuiteQL</code> object using <code>Query.toSuiteQL()</code>, you cannot modify this property.</p>
-----------------------------	--

Type	Array<string   number   boolean> (read-only)
Module	N/query Module
Parent Object	query.SuiteQL
Sibling Object Members	SuiteQL Object Members
Since	2020.1

## Syntax



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

TBD

## SuiteQL.query



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>The string representation of the SuiteQL query.</p> <p>This string can contain the following types of elements:</p> <ul style="list-style-type: none"> <li>■ SQL clauses, such as <code>SELECT</code>, <code>FROM</code>, and <code>WHERE</code></li> <li>■ Record or table names, such as <code>customer</code></li> <li>■ Field names using dot notation, such as <code>customer.entityid</code></li> <li>■ Operators for conditions, such as <code>=</code> and <code>&gt;=</code></li> <li>■ Field formatting metadata, such as <code>/*{entityid#RAW}*/</code></li> <li>■ Formatting characters, such as newline characters (<code>\n</code>)</li> </ul> <p>This property is read-only. After you create a <code>query.SuiteQL</code> object using <code>Query.toSuiteQL()</code>, you cannot modify this property.</p>
Type	string (read-only)
Module	N/query Module
Parent Object	query.SuiteQL
Sibling Object Members	SuiteQL Object Members
Since	2020.1

## Syntax



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

TBD

## SuiteQL.type



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The type of the query.  This property uses values from the <a href="#">query.Type</a> enum. When you use <a href="#">Query.toSuiteQL()</a> to convert an existing <a href="#">query.Query</a> object to SuiteQL, this property contains the same type that was specified for the original query.  This property is read-only. After you create a <a href="#">query.SuiteQL</a> object using <a href="#">Query.toSuiteQL()</a> , you cannot modify this property.
<b>Type</b>	string (read-only)
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.SuiteQL</a>
<b>Sibling Object Members</b>	<a href="#">SuiteQL Object Members</a>
<b>Since</b>	2020.1

## Syntax



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

TBD

## SuiteQL.run()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Runs the SuiteQL query and returns the query results.  You can use this method to run the SuiteQL query and obtain the results as a <a href="#">query.ResultSet</a> object. If you want to run the SuiteQL query as a paged query, use <a href="#">SuiteQL.runPaged(options)</a> .  For more information and examples of using SuiteQL in the N/query module, see <a href="#">SuiteQL in the N/query Module</a> . For more information about SuiteQL in general, see the help topic <a href="#">SuiteQL</a> .
<b>Returns</b>	<a href="#">query.ResultSet</a>
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.SuiteQL</a>
<b>Sibling Object Members</b>	<a href="#">SuiteQL Object Members</a>

Since	2020.1
-------	--------

## Syntax



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

TBD

## SuiteQL.runPaged(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Runs the SuiteQL query as a paged query and returns the paged query results.  You can use this method to run the SuiteQL query and obtain the results as a <a href="#">query.PagedData</a> object. If you want to run the SuiteQL query as a non-paged query, use <a href="#">SuiteQL.run()</a> .  For more information and examples of using SuiteQL in the N/query module, see <a href="#">SuiteQL in the N/query Module</a> . For more information about SuiteQL in general, see the help topic <a href="#">SuiteQL</a> .
<b>Returns</b>	<a href="#">query.PagedData</a>
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/query Module</a>
<b>Parent Object</b>	<a href="#">query.SuiteQL</a>
<b>Sibling Object Members</b>	<a href="#">SuiteQL Object Members</a>
<b>Since</b>	2020.1

## Parameters



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

Parameter	Type	Required / Optional	Description
options.pageSize	number	optional	The size of each page in the query results. The default value is 50 results per page.

## Syntax



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

TBD

## query.create(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates a <code>query.Query</code> object.</p> <p>Use this method to create your initial query definition. The initial query definition uses one query type. For available query types, see <code>query.Type</code>.</p> <p>After you create the initial query definition, use <code>Query.autoJoin(options)</code> to create your first join. Then use <code>Query.autoJoin(options)</code> or <code>Component.autoJoin(options)</code> to create all subsequent joins.</p> <p>For standard record types, the query type that you specify is validated immediately and must be one of the values in the <code>query.Type</code> enum. For custom record types, the query type that you specify is not validated until the query is executed using <code>Query.run()</code> or <code>Query.runPaged()</code> (or using the promise versions of these methods). If you specify a query type for a custom record type that does not exist, this method allows you to create the query and does not throw an error. However, when you execute the query, an error is thrown.</p>
	<p><b>Important:</b> The N/query module lets you create and run queries using the SuiteAnalytics Workbook query engine. 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 <a href="#">Navigating SuiteAnalytics Workbook</a>.</p>
	For more information about creating queries, see <a href="#">Scripting with the N/query Module</a> .
<b>Returns</b>	<code>query.Query</code>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/query Module
<b>Sibling Module Members</b>	N/query Module Members
<b>Since</b>	2018.1

## Parameters



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

Parameter	Type	Required / Optional	Description
<code>options.type</code>	string	required	<p>The query type that you want to use for the initial query definition.</p> <p>Use the <code>query.Type</code> enum to set this value (for an example, see the help topic <a href="#">Syntax</a>).</p> <p>When you execute <code>query.create(options)</code>, the <code>Query.type</code> property is set based on this value.</p>

Parameter	Type	Required / Optional	Description
			<p><b>Important:</b> The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic <a href="#">Available Record Types</a>.</p>

## Errors

Error Code	Thrown If
INVALID_RCRD_TYPE	<p>The specified query type is invalid.</p> <p><b>Note:</b> This error is not thrown if you specify a custom record type as the query type. Custom record types are validated when the query is executed using <a href="#">Query.run()</a> or <a href="#">Query.runPaged()</a> (or using the promise versions of these methods).</p>

## Syntax

**Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'id'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'entityid'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'email'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'hiredate'
    })
];

myCustomerQuery.sort = [
```

```

myCustomerQuery.createSort({
    column: myCustomerQuery.columns[1]
}),
mySalesRepJoin.createSort({
    column: mySalesRepJoin.columns[0],
    ascending: false
})
];

var resultSet = myCustomerQuery.run();
...
// Add additional code

```

## query.createRelativeDate(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a <a href="#">query.RelativeDate</a> object that represents a date relative to the current date. Use this method to create a <a href="#">query.RelativeDate</a> object to use as part of a query condition. After you create a <a href="#">query.RelativeDate</a> object, you can use it directly in the <code>values</code> parameter of <code>Query.createCondition(options)</code> or <code>Component.createCondition(options)</code> . When you call this method, the <code>options.dateId</code> parameter determines the relative date that is created. The <code>options.dateId</code> parameter uses values from the <a href="#">query.DateId</a> enum, and these values describe potential dates relative to the current date. Use them along with the <code>options.value</code> parameter to create a relative date. For example, to create a relative date that represents the date three weeks before the current date, call <code>query.createRelativeDate(options)</code> with an <code>options.dateId</code> value of <code>query.DateId.WEEKS_AGO</code> and an <code>options.value</code> value of 3. To create a relative date that represents the date three weeks after the current date, call <code>query.createRelativeDate(options)</code> with an <code>options.dateId</code> value of <code>query.DateId.WEEKS_FROM_NOW</code> and an <code>options.value</code> value of 3.
<b>Returns</b>	<a href="#">query.RelativeDate</a>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/query Module</a>
<b>Sibling Module Members</b>	<a href="#">N/query Module Members</a>
<b>Since</b>	2019.1

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.dateId</code>	string	required	The ID of the relative date to create.

Parameter	Type	Required / Optional	Description
			Use the <code>query.DateId</code> enum to set this value.
<code>options.value</code>	number	required	The value to use to create the relative date.  This value depends on the value that you specify for <code>options.dateId</code> . For example, to create a relative date that represents the date five days before the current date, use an <code>options.value</code> value of 5 and an <code>options.dateId</code> value of <code>query.DateId.DAYS_AGO</code> .

## Syntax

 **Important:** The following code sample 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 myRelativeDate = query.createRelativeDate({
    dateId: query.DateId.DAYS_AGO,
    value: 5
});

myTransactionQuery.condition = myTransactionQuery.createCondition({
    fieldId: 'trandate',
    operator: query.Operator.WITHIN,
    values: [query.RelativeDateRange.THREE_FISCAL_YEARS_AGO.start, myRelativeDate]
});
...
// Add additional code
```

## query.delete(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Deletes an existing query.</p> <p>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.</p> <p> <b>Important:</b> The N/query module lets you create and run queries using the SuiteAnalytics Workbook query engine. 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 <a href="#">Navigating SuiteAnalytics Workbook</a>.</p>
<b>Returns</b>	void

<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/query Module</a>
<b>Sibling Module Members</b>	<a href="#">N/query Module Members</a>
<b>Since</b>	2018.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.id	string	required	The script ID of the query to delete.

## Errors

Error Code	Thrown If
UNABLE_TO_DELETE_QUERY	A query with the specified ID cannot be deleted because the query does not exist or you do not have permission to delete it.

## Syntax

 **Important:** The following code sample 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.delete({
    id: 'custworkbook237'
});
...
// Add additional code
```

## query.load(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Loads an existing query as a <a href="#">query.Query</a> 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 query types, and execute the query in the same way as queries that you create using <a href="#">query.create(options)</a> .
---------------------------	--

	 <b>Important:</b> The N/query module lets you create and run queries using the SuiteAnalytics Workbook query engine. 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 <a href="#">Navigating SuiteAnalytics Workbook</a> .
Returns	<a href="#">query.Query</a>
Supported Script Types	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Governance	5 units
Module	<a href="#">N/query Module</a>
Sibling Module Members	<a href="#">N/query Module Members</a>
Since	2018.2

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.id</code>	string	required	The script ID of the query to load.

## Errors

Error Code	Thrown If
<code>UNABLE_TO_LOAD_QUERY</code>	A query with the specified ID cannot be loaded because the query does not exist or you do not have permission to load it.

## Syntax

 **Important:** The following code sample 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 myLoadedQuery = query.load({
  id: 'custworkbook237'
});

var mySalesRepJoin = myLoadedQuery.autoJoin({
  fieldId: 'salesrep'
});

var results = myLoadedQuery.run();
...
// Add additional code
```

## query.load.promise(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Loads an existing query asynchronously as a <a href="#">query.Query</a> object.</p> <p>Use this method to asynchronously 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 query types, and execute the query in the same way as queries that you create using <a href="#">query.create(options)</a>.</p> <p><b>Important:</b> The N/query module lets you create and run queries using the SuiteAnalytics Workbook query engine. 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 <a href="#">Navigating SuiteAnalytics Workbook</a>.</p> <p><b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">query.load(options)</a>. For more information on promises, see <a href="#">Promise Object</a>.</p>
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">query.load(options)</a>
<b>Supported Script Types</b>	<p>Client and server scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/query Module</a>
<b>Sibling Module Members</b>	<a href="#">N/query Module Members</a>
<b>Since</b>	2018.2

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.id</code>	string	required	The script ID of the query to load.

## Errors

Error Code	Thrown If
UNABLE_TO_LOAD_QUERY	A query with the specified ID cannot be loaded because the query does not exist or you do not have permission to load it.

## query.runSuiteQL(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Runs an arbitrary SuiteQL query.</p> <p>SuiteQL is a query language based on the SQL-92 revision of the SQL database query language. It provides advanced query capabilities you can use to access your NetSuite records and data.</p> <p>You can specify the SuiteQL query as one of the following:</p> <ul style="list-style-type: none"> <li>■ A string representation of the SuiteQL query</li> </ul> <pre>var results = query.runSuiteQL({     query: 'SELECT customer.entityid, customer.email FROM customer' });</pre> <ul style="list-style-type: none"> <li>■ A <code>query.SuiteQL</code> object</li> </ul> <pre>// In this example, mySuiteQLCustomerQuery is an existing SuiteQL object var results = query.runSuiteQL(mySuiteQLCustomerQuery);</pre> <ul style="list-style-type: none"> <li>■ A JavaScript Object that contains a <code>query</code> property and, optionally, a <code>params</code> property</li> </ul> <pre>var results = query.runSuiteQL({     query: 'SELECT customer.entityid, customer.email FROM customer WHERE customer.isperson = ?',     params: [true] });</pre> <p>For more information and examples of using SuiteQL in the N/query module, see <a href="#">SuiteQL in the N/query Module</a>. For more information about SuiteQL in general, see the help topic <a href="#">SuiteQL</a>.</p>
<b>Returns</b>	<code>query.ResultSet</code>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	To be confirmed
<b>Module</b>	<a href="#">N/query Module</a>
<b>Sibling Module Members</b>	<a href="#">N/query Module Members</a>
<b>Since</b>	2020.1

## Parameters



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

Parameter	Type	Required / Optional	Description
<code>options.query</code>	string	required	The string representation of the SuiteQL query to run.

Parameter	Type	Required / Optional	Description
options.params	Array<string   number   boolean>	optional	The parameters to use in the SuiteQL query.

## Errors

Error Code	Thrown If
MISSING_REQD_ARGUMENT	The parameter is missing.
SSS_INVALID_TYPE_ARG	Types other than string, number, or boolean are included in the <code>options.params</code> array.

## Syntax



**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

TBD

## query.runSuiteQLPaged(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Runs an arbitrary SuiteQL query as a paged query.</p> <p>SuiteQL is a query language based on the SQL-92 revision of the SQL database query language. It provides advanced query capabilities you can use to access your NetSuite records and data.</p> <p>You can specify the SuiteQL query as one of the following:</p> <ul style="list-style-type: none"> <li>■ A string representation of the SuiteQL query</li> </ul> <pre>var results = query.runSuiteQLPaged({     query: 'SELECT customer.entityid, customer.email FROM customer',     pageSize: 10 });</pre> <ul style="list-style-type: none"> <li>■ A <code>query.SuiteQL</code> object</li> </ul> <pre>// In this example, mySuiteQLCustomerQuery is an existing SuiteQL object var pageSizeObject = {pageSize: 10}; var results = query.runSuiteQL(mySuiteQLCustomerQuery, ...pageSizeObject);</pre> <ul style="list-style-type: none"> <li>■ A JavaScript Object that contains a <code>query</code> property and, optionally, a <code>params</code> property</li> </ul> <pre>var results = query.runSuiteQL({     query: 'SELECT customer.entityid, customer.email FROM customer WHERE customer.isperson = ?',     params: [true] });</pre>
---------------------------	--

	<pre>params: [true], pageSize: 10 });</pre> <p>For more information and examples of using SuiteQL in the N/query module, see <a href="#">SuiteQL in the N/query Module</a>. For more information about SuiteQL in general, see the help topic <a href="#">SuiteQL</a>.</p>
<b>Returns</b>	<code>query.PagedData</code>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	To be confirmed
<b>Module</b>	<a href="#">N/query Module</a>
<b>Sibling Module Members</b>	<a href="#">N/query Module Members</a>
<b>Since</b>	2020.1

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.query</code>	string	required	The string representation of the SuiteQL query to run.
<code>options.params</code>	Array<string   number   boolean>	optional	The parameters to use in the SuiteQL query.
<code>options.pageSize</code>	number	optional	The size of each page in the query results. The default value is 50 results per page.

## Errors

Error Code	Thrown If
<code>MISSING_REQD_ARGUMENT</code>	The parameter is missing.
<code>SSS_INVALID_TYPE_ARG</code>	Types other than string, number, or boolean are included in the <code>options.params</code> array.

## Syntax

 **Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

TBD

## query.Aggregate



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	<p>Holds the string values for aggregate functions supported with the <a href="#">N/query Module</a>. An aggregate function performs a calculation on the column or condition values and returns a single value.</p> <p>Each value in this enum (except <b>MEDIAN</b>) has two variants: distinct (using the <b>_DISTINCT</b> 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 <b>MAXIMUM</b> aggregate function is used to determine the maximum of a set of values. When using the distinct variant (<b>MAXIMUM_DISTINCT</b>), 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 (<b>MAXIMUM</b>), only one instance of the maximum value is listed, regardless of the number of instances of that maximum value in the set.</p> <p>This enum is used to pass the aggregate function argument to <a href="#">Component.createColumn(options)</a>, <a href="#">Component.createCondition(options)</a>, <a href="#">Query.createColumn(options)</a>, and <a href="#">Query.createCondition(options)</a>.</p> <p><b>Note:</b> 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>
<b>Module</b>	<a href="#">N/query Module</a>
<b>Sibling Module Members</b>	<a href="#">N/query Module Members</a>
<b>Since</b>	2018.1

## Values

Value	Description
AVERAGE	Calculates the average value.
AVERAGE_DISTINCT	Calculates the average distinct value.
COUNT	Counts the number of results.
COUNT_DISTINCT	Counts the number of distinct results.
MAXIMUM	Determines the maximum value. If the values are dates, the most recent date is determined.
MAXIMUM_DISTINCT	Determines the maximum distinct value. If the values are dates, the most recent date is determined.
MEDIAN	Calculates the median value.
MINIMUM	Determines the minimum value. If the values are dates, the earliest date is determined.
MINIMUM_DISTINCT	Determines the minimum distinct value. If the values are dates, the earliest date is determined.

Value	Description
SUM	Adds all values.
SUM_DISTINCT	Adds all distinct values.

## Syntax

 **Important:** The following code sample 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 myAggColumn = myTransactionQuery.createColumn({
    fieldId: 'amount',
    aggregate: query.Aggregate.AVERAGE
});

myTransactionQuery.columns = [myAggColumn];
...
// Add additional code
```

## query.DateId



**Note:** The content in this help topic pertains to SuiteScript 2.0.

Enum Description	Holds the string values for supported date codes in relative dates.  This enum is used to pass the date ID argument to <a href="#">query.createRelativeDate(options)</a> . It is also used as the value of the <a href="#">RelativeDate.dateId</a> property. When <a href="#">query.createRelativeDate(options)</a> is called, the enum value that you specify is set as the value of the <a href="#">RelativeDate.dateId</a> property.  When creating a relative date using <a href="#">query.createRelativeDate(options)</a> , use the values in this enum to specify a date relative to the current date. For example, to create a relative date that represents the date a certain number of days before the current date, use the <a href="#">DateId.DAYS_AGO</a> enum value. To create a relative date that represents the date a certain number of months after the current date, use the <a href="#">DateId.MONTHS_FROM_NOW</a> enum value.  The values in this enum might look similar to the values in the <a href="#">query.RelativeDateRange</a> enum, but each enum is used for a different purpose: <ul style="list-style-type: none"> <li>■ Use <a href="#">query.DateId</a> enum values to create a <a href="#">query.RelativeDate</a> object using <a href="#">query.createRelativeDate(options)</a>. After you create this object, you can use it in query conditions that you create using <a href="#">Query.createCondition(options)</a> or <a href="#">Component.createCondition(options)</a>.</li> <li>■ Use <a href="#">query.RelativeDateRange</a> enum values directly in query conditions that you create using <a href="#">Query.createCondition(options)</a> or <a href="#">Component.createCondition(options)</a>. Each value in the <a href="#">query.RelativeDateRange</a> enum represents a date range, and you can use these values in the <a href="#">values</a> parameter of <a href="#">Query.createCondition(options)</a> or <a href="#">Component.createCondition(options)</a>.</li> </ul>
------------------	--

	<p> <b>Note:</b> 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>
<b>Module</b>	N/query Module
<b>Sibling Module Members</b>	N/query Module Members
<b>Since</b>	2019.1

## Values

Value	Sets RelativeDate.dateId Property To
DAYs_Ago	dago
DAYs_From_Now	dfn
HOURs_Ago	hago
HOURs_From_Now	hfn
MINUTEs_Ago	nago
MINUTEs_From_Now	fn
MONTHs_Ago	mago
MONTHs_From_Now	mfn
QUARTERs_Ago	qago
QUARTERs_From_Now	qfn
SECONDs_Ago	sago
SECONDs_From_Now	sfn
WEEKs_Ago	wago
WEEKs_From_Now	wfn
YEARs_Ago	yago
YEARs_From_Now	yfn

## Syntax



**Important:** The following code sample 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 myRelativeDate = query.createRelativeDate({
    dateId: query.DateId.DAYs_AGO,
    value: 2
});
```

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

## query.FieldContext

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the string values for the field context to use when creating a column using <a href="#">Query.createColumn(options)</a> or <a href="#">Component.createColumn(options)</a> .  The field context determines how field values are displayed in a column. For example, you can specify that a column should display raw data (such as internal IDs), consolidated or converted amounts (such as currency totals), or user-friendly values (such as names).   <b>Note:</b> 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.
<b>Module</b>	<a href="#">N/query Module</a>
<b>Sibling Module Members</b>	<a href="#">N/query Module Members</a>
<b>Since</b>	2019.1

## Values

Value	Description
CONVERTED	Displays converted currency amounts using the exchange rate that was in effect on a specific date.
CURRENCY_CONSOLIDATED	Displays consolidated currency amounts in the base currency.
DISPLAY	Displays user-friendly field values. For example, for the entity field on Transaction records, using the DISPLAY enum value displays the name of the entity instead of its ID.
HIERARCHY	Displays user-friendly field values for hierarchical fields (for example, "Parent Company : SUB CAD"). This value is similar to the DISPLAY enum value but applies to hierarchical fields.
HIERARCHY_IDENTIFIER	Displays raw field values for hierarchical fields (for example, "1 : 5"). This value is similar to the RAW enum value but applies to hierarchical fields.
RAW	Displays raw field values. For example, for the entity field on Transaction records, using the RAW enum value displays the ID of the entity.

## Syntax

 **Important:** The following code sample 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 myContextColumn = myTransactionQuery.createColumn({
    fieldId: 'netamount',
    context: query.FieldContext.CURRENCY_CONSOLIDATED
});
...
// Add additional code

```

## query.Operator



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the string values for operators supported with the N/query Module. This enum is used to pass the operator argument to <a href="#">Query.createCondition(options)</a> and <a href="#">Component.createCondition(options)</a> .
	<p><b>Note:</b> 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>
<b>Module</b>	<a href="#">N/query Module</a>
<b>Sibling Module Members</b>	<a href="#">N/query Module Members</a>
<b>Since</b>	2018.1

## Values

Value
AFTER
AFTER_NOT
ANY_OF
ANY_OF_NOT
BEFORE
BEFORE_NOT
BETWEEN
BETWEEN_NOT
CONTAIN
CONTAIN_NOT
EMPTY

Value
EMPTY_NOT
ENDWITH
ENDWITH_NOT
EQUAL
EQUAL_NOT
GREATER
GREATER_NOT
GREATER_OR_EQUAL
GREATER_OR_EQUAL_NOT
IS
IS_NOT
LESS
LESS_NOT
LESS_OR_EQUAL
LESS_OR_EQUAL_NOT
ON
ON_NOT
ON_OR_AFTER
ON_OR_AFTER_NOT
ON_OR_BEFORE
ON_OR_BEFORE_NOT
START_WITH
START_WITH_NOT
WITHIN
WITHIN_NOT

## Syntax



**Important:** The following code sample 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
});
```

```

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

var firstCondition = myCustomerQuery.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var secondCondition = myCustomerQuery.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var thirdCondition = mySalesRepJoin.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});

myCustomerQuery.condition = myCustomerQuery.and(
    thirdCondition, myCustomerQuery.not(
        myCustomerQuery.or(firstCondition, secondCondition)
    )
);
var resultSet = myCustomerQuery.run();
...
// Add additional code

```

## query.RelativeDateRange



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	<p>Holds <code>query.RelativeDate</code> object values for supported date ranges in relative dates.</p> <p>This enum is used to pass the values argument to <code>Query.createCondition(options)</code> and <code>Component.createCondition(options)</code>. It is also used as the value of the <code>RelativeDate.value</code> property. Each value in this enum represents a date range. When <code>Query.createCondition(options)</code> or <code>Component.createCondition(options)</code> is called with a <code>query.RelativeDate</code> object as the <code>values</code> argument, this object is set as the value of the <code>RelativeDate.value</code> property.</p> <p>When creating a condition using <code>Query.createCondition(options)</code> or <code>Component.createCondition(options)</code>, use the values in this enum (along with values in the <code>query.Operator</code> 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 date, use the <code>query.RelativeDateRange.TODAY</code> enum value and the <code>query.Operator.BEFORE</code> enum value. To create a condition to match dates that occur after last year, use the <code>query.RelativeDateRange.LAST_YEAR</code> enum value and the <code>query.Operator.AFTER</code> enum value. For more information about relative dates, see <a href="#">Relative Dates in the N/query Module</a>.</p> <p>The values in this enum might look similar to the values in the <code>query.DateId</code> enum, but each enum is used for a different purpose:</p> <ul style="list-style-type: none"> <li>■ Use <code>query.DateId</code> enum values to create a <code>query.RelativeDate</code> object using <code>query.createRelativeDate(options)</code>. After you create this object, you can use it</li> </ul>
-------------------------	---

	<p>in query conditions that you create using <a href="#">Query.createCondition(options)</a> or <a href="#">Component.createCondition(options)</a>.</p> <ul style="list-style-type: none"> <li>■ Use <code>query.RelativeDateRange</code> enum values directly in query conditions that you create using <a href="#">Query.createCondition(options)</a> or <a href="#">Component.createCondition(options)</a>. Each value in the <code>query.RelativeDateRange</code> enum represents a date range, and you can use these values in the <code>values</code> parameter of <a href="#">Query.createCondition(options)</a> or <a href="#">Component.createCondition(options)</a>.</li> </ul> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p><b>Note:</b> 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> </div>
<b>Module</b>	N/query Module
<b>Sibling Module Members</b>	<a href="#">N/query Module Members</a>
<b>Since</b>	2019.1

## Values

Value	RelativeDate.dateId Property
FISCAL_HALF_BEFORE_LAST	FHBL
FISCAL_HALF_BEFORE_LAST_TO_DATE	FHBLTD
FISCAL_QUARTER_BEFORE_LAST	FQBL
FISCAL_QUARTER_BEFORE_LAST_TO_DATE	FQBLTD
FISCAL_YEAR_BEFORE_LAST	FYBL
FISCAL_YEAR_BEFORE_LAST_TO_DATE	FYBLTD
FIVE_DAYS_AGO	DAGO5
FIVE_DAYS_FROM_NOW	DFN5
FOUR_DAYS_AGO	DAGO4
FOUR_DAYS_FROM_NOW	DFN4
FOUR_WEEKS_STARTING_THIS_WEEK	TWN3W
LAST_BUSINESS_WEEK	LBW
LAST_FISCAL_HALF	LFH
LAST_FISCAL_HALF_ONE_FISCAL_YEAR_AGO	LFHLFY
LAST_FISCAL_HALF_TO_DATE	LFHTD
LAST_FISCAL_QUARTER	LFQ
LAST_FISCAL_QUARTER_ONE_FISCAL_YEAR_AGO	LFQLFY
LAST_FISCAL_QUARTER_TO_DATE	LFQTD
LAST_FISCAL_QUARTER_TWO_FISCAL_YEARS_AGO	LFQFYBL

Value	RelativeDate.dateId Property
LAST_FISCAL_YEAR	LFY
LAST_FISCAL_YEAR_TO_DATE	LFYTD
LAST_MONTH	LM
LAST_MONTH_ONE_FISCAL_QUARTER_AGO	LMLFQ
LAST_MONTH_ONE_FISCAL_YEAR_AGO	LMLFY
LAST_MONTH_TO_DATE	LMTD
LAST_MONTH_TWO_FISCAL_QUARTERS_AGO	LMFQBL
LAST_MONTH_TWO_FISCAL_YEARS_AGO	LMFYBL
LAST_ROLLING_HALF	LRH
LAST_ROLLING_QUARTER	LRQ
LAST_ROLLING_YEAR	LRY
LAST_WEEK	LW
LAST_WEEK_TO_DATE	LWTD
LAST_YEAR	LY
LAST_YEAR_TO_DATE	LYTD
MONTH_AFTER_NEXT	MAN
MONTH_AFTER_NEXT_TO_DATE	MANTD
MONTH_BEFORE_LAST	MBL
MONTH_BEFORE_LAST_TO_DATE	MBLTD
NEXT_BUSINESS_WEEK	NBW
NEXT_FISCAL_HALF	NFH
NEXT_FISCAL_QUARTER	NFQ
NEXT_FISCAL_YEAR	NFY
NEXT_FOUR_WEEKS	N4W
NEXT_MONTH	NM
NEXT_ONE_HALF	NOH
NEXT_ONE_MONTH	NOM
NEXT_ONE_QUARTER	NOQ
NEXT_ONE_WEEK	NOW
NEXT_ONE_YEAR	NOY
NEXT_WEEK	NW
NINETY_DAYS_AGO	DAGO90

Value	RelativeDate.dateId Property
NINETY_DAYS_FROM_NOW	DFN90
ONE_YEAR_BEFORE_LAST	OYBL
PREVIOUS_FISCAL_QUARTERS_LAST_FISCAL_YEAR	PQLFY
PREVIOUS_FISCAL_QUARTERS_THIS_FISCAL_YEAR	PQTFY
PREVIOUS_MONTHS_LAST_FISCAL_HALF	PMLFH
PREVIOUS_MONTHS_LAST_FISCAL_QUARTER	PMLFQ
PREVIOUS_MONTHS_LAST_FISCAL_YEAR	PMLFY
PREVIOUS_MONTHS_SAME_FISCAL_HALF_LAST_FISCAL_YEAR	PMSFHLFY
PREVIOUS_MONTHS_SAME_FISCAL_QUARTER_LAST_FISCAL_YEAR	PMSFQLFY
PREVIOUS_MONTHS_THIS_FISCAL_HALF	PMTFH
PREVIOUS_MONTHS_THIS_FISCAL_QUARTER	PMTFQ
PREVIOUS_MONTHS_THIS_FISCAL_YEAR	PMTFY
PREVIOUS_ONE_DAY	OD
PREVIOUS_ONE_HALF	OH
PREVIOUS_ONE_MONTH	OM
PREVIOUS_ONE_QUARTER	OQ
PREVIOUS_ONE_WEEK	OW
PREVIOUS_ONE_YEAR	OY
PREVIOUS_ROLLING_HALF	PRH
PREVIOUS_ROLLING_QUARTER	PRQ
PREVIOUS_ROLLING_YEAR	PRY
SAME_DAY_FISCAL_QUARTER_BEFORE_LAST	SDFQBL
SAME_DAY_FISCAL_YEAR_BEFORE_LAST	SDFYBL
SAME_DAY_LAST_FISCAL_QUARTER	SDLFQ
SAME_DAY_LAST_FISCAL_YEAR	SDLFY
SAME_DAY_LAST_MONTH	SDLM
SAME_DAY_LAST_WEEK	SDLW
SAME_DAY_MONTH_BEFORE_LAST	SDMBL
SAME_DAY_WEEK_BEFORE_LAST	SDWBL
SAME_FISCAL_HALF_LAST_FISCAL_YEAR	SFHLFY
SAME_FISCAL_HALF_LAST_FISCAL_YEAR_TO_DATE	SFHLFYTD
SAME_FISCAL_QUARTER_FISCAL_YEAR_BEFORE_LAST	SFQFYBL

Value	RelativeDate.dateId Property
SAME_FISCAL_QUARTER_LAST_FISCAL_YEAR	SFQLFY
SAME_FISCAL_QUARTER_LAST_FISCAL_YEAR_TO_DATE	SFQLFYTD
SAME_MONTH_FISCAL_QUARTER_BEFORE_LAST	SMFQBL
SAME_MONTH_FISCAL_YEAR_BEFORE_LAST	SMFYBL
SAME_MONTH_LAST_FISCAL_QUARTER	SMLFQ
SAME_MONTH_LAST_FISCAL_QUARTER_TO_DATE	SMLFQTD
SAME_MONTH_LAST_FISCAL_YEAR	SMLFY
SAME_MONTH_LAST_FISCAL_YEAR_TO_DATE	SMLFYTD
SAME_WEEK_FISCAL_YEAR_BEFORE_LAST	SWFYBL
SAME_WEEK_LAST_FISCAL_YEAR	SWLFY
SIXTY_DAYS_AGO	DAGO60
SIXTY_DAYS_FROM_NOW	DFN60
TEN_DAYS_AGO	DAGO10
TEN_DAYS_FROM_NOW	DFN10
THIRTY_DAYS_AGO	DAGO30
THIRTY_DAYS_FROM_NOW	DFN30
THIS_BUSINESS_WEEK	TBW
THIS_FISCAL_HALF	TFH
THIS_FISCAL_HALF_TO_DATE	TFHTD
THIS_FISCAL_QUARTER	TFQ
THIS_FISCAL_QUARTER_TO_DATE	TFQTD
THIS_FISCAL_YEAR	TFY
THIS_FISCAL_YEAR_TO_DATE	TFYTD
THIS_MONTH	TM
THIS_MONTH_TO_DATE	TMTD
THIS_ROLLING_HALF	TRH
THIS_ROLLING_QUARTER	TRQ
THIS_ROLLING_YEAR	TRY
THIS_WEEK	TW
THIS_WEEK_TO_DATE	TWTD
THIS_YEAR	TY
THIS_YEAR_TO_DATE	TYTD

Value	RelativeDate.dateId Property
THREE_DAYS_AGO	DAGO3
THREE_DAYS_FROM_NOW	DFN3
THREE_FISCAL_QUARTERS_AGO	FQB
THREE_FISCAL_QUARTERS_AGO_TO_DATE	FQBTD
THREE_FISCAL_YEARS_AGO	FYB
THREE_FISCAL_YEARS_AGO_TO_DATE	FYBTD
THREE_MONTHS_AGO	MB
THREE_MONTHS_AGO_TO_DATE	MBTD
TODAY	TODAY
TODAY_TO_END_OF_THIS_MONTH	TODAYTTM
TOMORROW	TOMORROW
TWO_DAYS_AGO	DAGO2
TWO_DAYS_FROM_NOW	DFN2
WEEK_AFTER_NEXT	WAN
WEEK_AFTER_NEXT_TO_DATE	WANTD
WEEK_BEFORE_LAST	WBL
WEEK_BEFORE_LAST_TO_DATE	WBLTD
YESTERDAY	YESTERDAY

## Syntax



**Important:** The following code sample 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
});

myTransactionQuery.condition = myTransactionQuery.createCondition({
    fieldId: 'trandate',
    operator: query.Operator.BEFORE,
    values: query.RelativeDateRange.TODAY
});
...
// Add additional code
```

## query.ReturnType



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the string values for the formula return types supported with the <a href="#">N/query Module</a> . This enum is used to pass the formula return type argument to <a href="#">Query.createColumn(options)</a> , <a href="#">Component.createColumn(options)</a> , <a href="#">Query.createCondition(options)</a> , and <a href="#">Component.createCondition(options)</a> . For more information on formulas, see the help topics <a href="#">SuiteAnalytics Workbook Overview</a> , <a href="#">SQL Expressions</a> , and <a href="#">Search Formula Examples and Tips</a> .
<b>Module</b>	<a href="#">N/query Module</a>
<b>Sibling Module Members</b>	<a href="#">N/query Module Members</a>
<b>Since</b>	2018.1

## Values

Value
ANY
BOOLEAN
CURRENCY
DATE
DATETIME
DURATION
FLOAT
INTEGER
KEY
RELATIONSHIP
STRING
UNKNOWN

## Syntax



**Important:** The following code sample 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'
});
...
// Add additional code
```

## query.SortLocale



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the string values for sort locales supported with the <a href="#">N/query Module</a> . This enum is used to pass the locale argument to <a href="#">Query.createSort(options)</a> and <a href="#">Component.createSort(options)</a> .
	<p><b>Note:</b> 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>
<b>Module</b>	<a href="#">N/query Module</a>
<b>Sibling Module Members</b>	<a href="#">N/query Module Members</a>
<b>Since</b>	2018.2

## Values

Value
ARABIC
ARABIC_ABJ_MATCH
ARABIC_ABJ_MATCH_CI
ARABIC_ABJ_SORT
ARABIC_ABJ_SORT_CI
ARABIC_CI
ARABIC_MATCH

Value
ARABIC_MATCH_CI
ASCII7
ASCII7_CI
AZERBAIJANI
AZERBAIJANI_CI
BENGALI
BENGALI_CI
BIG5
BIG5_CI
BINARY
BINARY_CI
BULGARIAN
BULGARIAN_CI
CANADIAN_M
CATALAN
CATALAN_CI
CROATIAN
CROATIAN_CI
CS_CZ
CZECH
CZECH_CI
CZECH_PUNCTUATION
CZECH_PUNCTUATION_CI
DANISH
DANISH_CI
DANISH_M
DA_DK
DE_DE
DUTCH
DUTCH_CI
EBCDIC
EBCDIC_CI

Value
EEC_EURO
EEC_EUROPA3
EEC_EUROPA3_CI
EEC_EURO_CI
EN
EN_AU
EN_CA
EN_GB
EN_US
ESTONIAN
ESTONIAN_CI
ES_AR
ES_ES
FINNISH
FINNISH_CI
FRENCH
FRENCH_AI
FRENCH_CI
FRENCH_M
FR_CA
FR_FR
GBK
GBK_AI
GBK_CI
GENERIC_M
GERMAN
GERMAN_AI
GERMAN_CI
GERMAN_DIN
GERMAN_DIN_AI
GERMAN_DIN_CI
GREEK

Value
GREEK_AI
GREEK_CI
HEBREW
HEBREW_AI
HEBREW_CI
HE_IL
HKSCS
HKSCS_AI
HKSCS_CI
HUNGARIAN
HUNGARIAN_AI
HUNGARIAN_CI
ICELANDIC
ICELANDIC_AI
ICELANDIC_CI
ID_ID
INDONESIAN
INDONESIAN_AI
INDONESIAN_CI
ITALIAN
ITALIAN_AI
ITALIAN_CI
IT_IT
JAPANESE_M
JA_JP
KOREAN_M
KO_KR
LATIN
LATIN_AI
LATIN_CI
LATVIAN
LATVIAN_AI

Value
LATVIAN_CI
LITHUANIAN
LITHUANIAN_AI
LITHUANIAN_CI
MALAY
MALAY_AI
MALAY_CI
NL_NL
NO_NO
NORWEGIAN
NORWEGIAN_AI
NORWEGIAN_CI
POLISH
POLISH_AI
POLISH_CI
PT_BR
PUNCTUATION
PUNCTUATION_AI
PUNCTUATION_CI
ROMANIAN
ROMANIAN_AI
ROMANIAN_CI
RUSSIAN
RUSSIAN_AI
RUSSIAN_CI
RU_RU
SCHINESE_PINYIN_M
SCHINESE_RADICAL_M
SCHINESE_STROKE_M
SLOVAK
SLOVAK_AI
SLOVAK_CI

Value
SLOVENIAN
SLOVENIAN_AI
SLOVENIAN_CI
SPANISH
SPANISH_AI
SPANISH_CI
SPANISH_M
SV_SE
SWEDISH
SWEDISH_AI
SWEDISH_CI
SWISS
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

Value
WEST_EUROPEAN_AI
WEST_EUROPEAN_CI
ZH_CN
ZH_TW

## Syntax

**⚠ Important:** The following code sample 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'
    })
];

myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[0],
        locale: query.SortLocale.EN_CA
    })
];
...
// Add additional code
```

## query.Type

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the string values for query types used in the query definition. This enum is used to pass the initial query type argument to <a href="#">query.create(options)</a> . <p><b>⚠ Important:</b> The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic <a href="#">Available Record Types</a>.</p> <p><b>i Note:</b> 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>
<b>Module</b>	N/query Module

<b>Sibling Module Members</b>	N/query Module Members
<b>Since</b>	2018.1

## Values

<b>Note:</b> Before using these values, consider the following:
<ul style="list-style-type: none"> <li>■ A query type is not the same as a record type. The supported query types listed below do not necessarily correspond with the supported record types listed in the <a href="#">N/record Module</a>.</li> <li>■ Depending on your account, role, and enabled features, some of these values may not be available.</li> <li>■ Custom record types are not included in this enum.</li> </ul>

Enum Value	Sets Query.type Property To
ACCOUNT	account
ACCOUNTING_BOOK	accountingbook
ACCOUNTING_CONTEXT	accountingcontext
ACCOUNTING_PERIOD	accountingperiod
ALLOCATION_METHOD	allocationmethod
AMORTIZATION_SCHEDULE	amortizationschedule
AMORTIZATION_TEMPLATE	amortizationtemplate
BILLING_CLASS	billingclass
BILLING_RATE_CARD	billingratecard
BILLING_SCHEDULE	billingschedule
BILL_OF_DISTRIBUTION	billofdistribution
BILL_RUN	billrun
BILL_RUN_SCHEDULE	billrunschedule
BIN	bin
BOM	bom
BOM_REVISION	bomrevision
BOM_REVISION_COMPONENT	bomrevisioncomponent
BUDGETCATEGORY	budgetcategory
BUDGETS	budgets
BUDGET_EXCHANGE_RATE	budgetexchangerate
BULK_PROC_SUBMISSION	bulkprobsubmission
BUNDLE_INSTALLATION_SCRIPT	bundleinstallationscript

Enum Value	Sets Query.type Property To
BUNDLE_INSTALLATION_SCRIPT_DEPLOYMENT	bundleinstallationscriptdeployment
BUSINESS_EVENTS_PROCESSING_HISTORY	businesseventsprocessinghistory
CALENDAR_EVENT	calendarevent
CAMPAIGN_AUDIENCE	campaignaudience
CAMPAIGN_CATEGORY	campaigncategory
CAMPAIGN_CHANNEL	campaignchannel
CAMPAIGN_EMAIL_ADDRESS	campaignemailaddress
CAMPAIGN_EVENT	campaignevent
CAMPAIGN_FAMILY	campaignfamily
CAMPAIGN_OFFER	campaignoffer
CAMPAIGN_RESPONSE	campaignresponse
CAMPAIGN_SEARCH_ENGINE	campaignsearchengine
CAMPAIGN_SUBSCRIPTION	campaignsubscription
CAMPAIGN_TEMPLATE	campaigntemplate
CAMPAIGN_VERTICAL	campaignvertical
CATEGORY1099MISC	category1099misc
CHARGE	charge
CHARGE_RULE	chargerule
CHARGE_RUN	chargerun
CHARGE_TYPE	chargetype
CLASSIFICATION	classification
CLIENT_SCRIPT	clientscript
CLIENT_SCRIPT_DEPLOYMENT	clientscriptdeployment
COMPETITOR	competitor
CONSOLIDATED_EXCHANGE_RATE	consolidatedexchangerate
CONSOLIDATED_RATE_ADJUSTOR_PLUGIN	consolidatedrateadjustorplugin
CONTACT	contact
CONTACT_CATEGORY	contactcategory
CONTACT_ROLE	contactrole
CONTACT_SUBSIDIARY_RELATIONSHIP	contactsubsidiaryrelationship
COST_CATEGORY	costcategory
COUPON_CODE	couponcode

Enum Value	Sets Query.type Property To
CURRENCY	currency
CURRENCY_RATE	currencyrate
CUSTOMER	customer
CUSTOMER_CATEGORY	customercategory
CUSTOMER_MESSAGE	customermessage
CUSTOMER_SEGMENT	customersegment
CUSTOMER_SUBSIDIARY_RELATIONSHIP	customersubsidiaryrelationship
CUSTOM_FIELD	customfield
CUSTOM_GL_PLUGIN	customglplugin
CUSTOM_LIST	customlist
CUSTOM_RECORD_TYPE	customrecordtype
CUSTOM_SEGMENT	customsegment
CUSTOM_TRANSACTION_TYPE	customtransactiontype
DELETED_RECORD	deletedrecord
DEPARTMENT	department
DEVICE_ID	deviceid
DISTRIBUTION_CATEGORY	distributioncategory
DISTRIBUTION_NETWORK	distributionnetwork
DOMAIN	domain
EMAIL_CAPTURE_PLUGIN	emailcaptureplugin
EMAIL_TEMPLATE	emailtemplate
EMPLOYEE	employee
EMPLOYEE_LIST	employeelist
EMPLOYEE_STATUS	employeeestatus
EMPLOYEE_SUBSIDIARY_RELATIONSHIP	employeesubsidiaryrelationship
EMPLOYEE_TYPE	employeetype
ENTITY	entity
ENTITY_GROUP	entitygroup
ENTITY_SUBSIDIARY_RELATIONSHIP	entitysubsidiaryrelationship
EXPENSE_CATEGORY	expensemecategory
FAX_TEMPLATE	faxtemplate
FILE	file

Enum Value	Sets Query.type Property To
FISCAL_CALENDAR	fiscalcalendar
FORECAST	forecast
FULFILLMENT_EXCEPTION_REASON	fulfillmentexceptionreason
FULFILLMENT_REQUEST	fulfillmentrequest
GATEWAY_NOTIFICATION	gatewaynotification
GENERAL_ALLOCATION_SCHEDULE	generalallocationschedule
GENERAL_TOKEN	generaltoken
GENERIC_RESOURCE	genericresource
GENERIC_RESOURCE_SUBSIDIARY_RELATIONSHIP	genericresourcesubsidiaryrelationship
GIFT_CERTIFICATE	giftcertificate
GLOBAL_ACCOUNT_MAPPING	globalaccountmapping
GLOBAL_INVENTORY_RELATIONSHIP	globalinventoryrelationship
GL_LINES_AUDIT_LOG	gllinesauditlog
GL_LINES_PLUGIN_REVISION	gllinespluginrevision
G_L_NUMBERING_SEQUENCE	glnumberingsequence
INBOUND_SHIPMENT	inboundshipment
INCOTERM	incoterm
INVENTORY_COST_TEMPLATE	inventorycosttemplate
INVENTORY_NUMBER	inventorynumber
INVENTORY_STATUS	inventorystatus
INVT_ITEM_PRICE_HISTORY	invitempricehistory
ISSUE	issue
ITEM	item
ITEM_ACCOUNT_MAPPING	itemaccountmapping
ITEM_COLLECTION	itemcollection
ITEM_DEMAND_PLAN	itemdemandplan
ITEM_LOCATION_CONFIGURATION	itemlocationconfiguration
ITEM_SEGMENT_INCLUDING_SYNTHETIC	itemsegmentincludingsynthetic
ITEM_SUPPLY_PLAN	itemsupplyplan
I_P_RESTRICTIONS	iprestrictions
JOB	job
JOB_RESOURCE_ROLE	jobresourcerole

Enum Value	Sets Query.type Property To
JOB_STATUS	jobstatus
JOB_TYPE	jobtype
KNOWLEDGE_BASE	knowledgebase
LOCATION	location
LOCATION_COSTING_GROUP	locationcostinggroup
LOGIN_AUDIT	loginaudit
MAIL_TEMPLATE	mailtemplate
MANUFACTURING_COST_TEMPLATE	manufacturingcosttemplate
MANUFACTURING_ROUTING	manufacturingrouting
MANUFACTURING_TRANSACTION	manufacturingtransaction
MAP_REDUCE_SCRIPT	mapreducescript
MAP_REDUCE_SCRIPT_DEPLOYMENT	mapreducescriptdeployment
MASS_UPDATE_SCRIPT	massupdatescript
MASS_UPDATE_SCRIPT_DEPLOYMENT	massupdatescriptdeployment
MEDIA_ITEM_FOLDER	mediaitemfolder
MEM_DOC	memdoc
MEM_DOC_TRANSACTION_TEMPLATE	memdoctransactiontemplate
MERCHANDISE_HIERARCHY_LEVEL	merchandisehierarchylevel
MERCHANDISE_HIERARCHY_NODE	merchandisehierarchynode
MERCHANDISE_HIERARCHY_VERSION	merchandisehierarchyversion
MESSAGE	message
MFG_PLANNED_TIME	mfgplannedtime
NEXUS	nexus
NOTE	note
ONLINE_CASE_FORM	onlinecaseform
ONLINE_FORM_TEMPLATE	onlineformtemplate
ONLINE_LEAD_FORM	onlineleadform
ORDER_ALLOCATION_STRATEGY	orderallocationstrategy
OTHER_NAME	othername
OTHER_NAME_CATEGORY	othernamecategory
OTHER_NAME_SUBSIDIARY_RELATIONSHIP	othernamesubsidiaryrelationship
OUTBOUND_REQUEST	outboundrequest

Enum Value	Sets Query.type Property To
O_AUTH_TOKEN	oauthtoken
PARTNER	partner
PARTNER_SUBSIDIARY_RELATIONSHIP	partnersubsidiaryrelationship
PAYCHECK	paycheck
PAYMENT_CARD_SEARCH_RECORD	paymentcardsearchrecord
PAYMENT_CARD_TOKEN	paymentcardtoken
PAYMENT_EVENT	paymentevent
PAYMENT_GATEWAY_PLUGIN	paymentgatewayplugin
PAYMENT_INSTRUMENT	paymentinstrument
PAYMENT_METHOD	paymentmethod
PAYMENT_PROCESSING_PROFILE	paymentprocessingprofile
PAYROLL_BATCH	payrollbatch
PAYROLL_ITEM	payrollitem
PDF_TEMPLATE	pdftemplate
PHONE_CALL	phonecall
PLANNED_STANDARD_COST	plannedstandardcost
PLATFORM_EXTENSION_PLUGIN	platformextensionplugin
PLUG_IN_TYPE	plugintype
PLUG_IN_TYPE_IMPL	plugintypeimpl
PORTLET	portlet
PORTLET_DEPLOYMENT	portletdeployment
PRICE_LEVEL	pricelevel
PRICING	pricing
PRICING_GROUP	pricinggroup
PROJECT_SUBSIDIARY_RELATIONSHIP	projectsubsidiaryrelationship
PROJECT_TASK	projecttask
PROJECT_TEMPLATE	projecttemplate
PROJECT_TEMPLATE_SUBSIDIARY_RELATIONSHIP	projecttemplatesubsidiaryrelationship
PROMOTIONS_PLUGIN	promotionsplugin
PROMOTION_CODE	promotioncode
PUBLISHED_SAVED_SEARCH	publishedsavedsearch
QUANTITY_PRICING_SCHEDULE	quantitypricingschedule

Enum Value	Sets Query.type Property To
QUOTA	quota
RECENT_RECORD	recentrecord
REDIRECT	redirect
RESOURCE_GROUP	resourcegroup
RESTLET	restlet
RESTLET_DEPLOYMENT	restletdeployment
REV_REC_SCHEDULE	revrecschedule
REV_REC_TEMPLATE	revrectemplate
ROLE	role
SALES_INVOICED	salesinvoiced
SALES_ORDERED	salesordered
SALES_TAX_ITEM	salestaxitem
SCHEDULED_SCRIPT	scheduledscript
SCHEDULED_SCRIPT_DEPLOYMENT	scheduledscriptdeployment
SCHEDULED_SCRIPT_INSTANCE	scheduledscriptinstance
SCRIPT	script
SCRIPT_DEPLOYMENT	scriptdeployment
SCRIPT_NOTE	scriptnote
SCRIPT_RECORD_TYPE	scriptrecordtype
SEARCH_CAMPAIGN	searchcampaign
SENT_EMAIL	sentemail
SHIPPING_PACKAGE	shippingpackage
SHIPPING_PARTNERS_PLUGIN	shippingpartnersplugin
SHIPPING_PARTNER_REGISTRATION	shippingpartnerregistration
SHIP_ITEM	shipitem
SHOPPING_CART	shoppingcart
SITE_CATEGORY	sitecategory
SOLUTION	solution
STANDARD_COST_VERSION	standardcostversion
STATISTICAL_JOURNAL_ENTRY	statisticaljournalentry
STATISTICAL_SCHEDULE	statisticalschedule
STORE_TAB	storetab

Enum Value	Sets Query.type Property To
SUBLIST	sublist
SUBSIDIARY	subsidiary
SUBSIDIARY_SETTINGS	subsidiarysettings
SUITELET	suitelet
SUITELET_DEPLOYMENT	suiteletdeployment
SUITE_SCRIPT_DETAIL	suitescriptdetail
SUPPLY_CHAIN_SNAPSHOT	supplychainsnapshot
SUPPLY_CHAIN_SNAPSHOT_SIMULATION	supplychainsnapshotsimulation
SUPPORT_CASE	supportcase
SYSTEM_EMAIL_TEMPLATE	systememailtemplate
SYSTEM_NOTE	systemnote
SYSTEM_NOTE2	systemnote2
SYSTEM_NOTE_FIELD	systemnotefield
TASK	task
TASK_ITEM_STATUS	taskitemstatus
TAX_CALCULATION_PLUGIN	taxcalculationplugin
TAX_TYPE	taxtype
TERM	term
TEST_PLUGIN	testplugin
TIME_BILL	timebill
TOPIC	topic
TRANSACTION	transaction
TRANSACTION_DELETION_REASON	transactiondeletionreason
TRANSACTION_HISTORY	transactionhistory
TRANSACTION_NUMBERING_AUDIT_LOG	transactionnumberingauditlog
UMD_FIELD	umdfield
UNDELIVERED_EMAIL	undeliveredemail
UNITS_TYPE	unitstype
USER_EVENT_SCRIPT	userevents
USER_EVENT_SCRIPT_DEPLOYMENT	userevents
USER_O_AUTH_TOKEN	useroauth
USRSAVEDSEARCH	usr

Enum Value	Sets Query.type Property To
USR_AUDIT_LOG	usrauditlog
USR_EXECUTION_LOG	usrexecutionlog
VENDOR	vendor
VENDOR_CATEGORY	vendorcategory
VENDOR_SUBSIDIARY_RELATIONSHIP	vendorsubsidiaryrelationship
WEBAPP	webapp
WEB_SITE	website
WORKFLOW_ACTION_SCRIPT	workflowactionscript
WORKFLOW_ACTION_SCRIPT_DEPLOYMENT	workflowactionscriptdeployment
WORKPLACE	workplace
WORK_CALENDAR	workcalendar

## Syntax



**Important:** The following code sample 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
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

var firstCondition = myCustomerQuery.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var secondCondition = myCustomerQuery.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var thirdCondition = mySalesRepJoin.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});

myCustomerQuery.condition = myCustomerQuery.and(
    thirdCondition, myCustomerQuery.or(firstCondition, secondCondition)
```

```

};

var resultSet = myCustomerQuery.run();
...
// Add additional code

```

## N/record Module

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">record.Column</a>	Object	Client and server-side scripts	Encapsulates a column of a sublist on a standard or custom record.
	<a href="#">record.Field</a>	Object	Client and server-side scripts	Encapsulates a body or sublist field on a standard or custom record.
	<a href="#">record.Macro</a>	Object	Client and server-side scripts	Encapsulates a NetSuite record macro.
	Plain JavaScript Object	Object	Client and server-side scripts	A plain JavaScript object of record macros available for a record type. This object is returned by <a href="#">Record.getMacros(options)</a> .
	<a href="#">record.Record</a>	Object	Client and server-side scripts	Encapsulates a NetSuite record.
	<a href="#">record.Sublist</a>	Object	Client and server-side scripts	Encapsulates a sublist on a standard or custom record.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">record.attach(options)</a>	void	Client and server-side scripts	Attaches a record to another record.
	<a href="#">record.attach.promise(options)</a>	Promise	Client scripts	Attaches a record asynchronously to another record.
	<a href="#">record.copy(options)</a>	<a href="#">record.Record</a>	Client and server-side scripts	Creates a new record by copying an existing record in NetSuite.
	<a href="#">record.copy.promise(options)</a>	Promise	Client scripts	Creates a new record asynchronously by copying an existing record in NetSuite.
	<a href="#">record.create(options)</a>	<a href="#">record.Record</a>	Client and server-side scripts	Creates a new record.
	<a href="#">record.create.promise(options)</a>	Promise	Client scripts	Creates a new record asynchronously.
	<a href="#">record.delete(options)</a>	number	Client and server-side scripts	Deletes a record.
	<a href="#">record.delete.promise(options)</a>	Promise	Client scripts	Deletes a record asynchronously.
	<a href="#">record.detach(options)</a>	void	Client and server-side scripts	Detaches a record from another record.
	<a href="#">record.detach.promise(options)</a>	Promise	Client scripts	Detaches a record from another record asynchronously.
	<a href="#">record.load(options)</a>	Object	Client and server-side scripts	Loads an existing record.
	<a href="#">record.submitFields(options)</a>	Object	Client and server-side scripts	Updates and submits one or more body fields on an existing record in NetSuite, and returns the internal ID of the parent record.
	<a href="#">record.submitFields.promise(options)</a>	Promise	Client scripts	Updates and submits one or more body fields asynchronously on an existing record in NetSuite, and returns the internal ID of the parent record.
	<a href="#">record.transform(options)</a>	<a href="#">record.Record</a>	Client and server-side scripts	Transforms a record from one type into another, using data from an existing record.
	<a href="#">record.transform.promise(options)</a>	Promise	Client scripts	Transforms a record from one type into another asynchronously, using data from an existing record.
Enum	<a href="#">record.Type</a>	enum	Client and server-side scripts	Enumeration that holds the string values for supported standard record types.

## Column Object Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	<a href="#">Column.id</a>	string (read-only)	Client and server-side scripts	Returns the internal ID of the column.
	<a href="#">Column.label</a>	string (read-only)	Client and server-side scripts	Returns the UI label for the column.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	Column.sublistId	string (read-only)	Client and server-side scripts	Returns the internal ID of the standard or custom sublist that contains the column.
	Column.type	string (read-only)	Client and server-side scripts	Returns the column type.

## Field Object Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Field.getSelectOptions(options)	array	Client and server-side scripts	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.
Property	Field.label	string (read-only)	Client and server-side scripts	Returns the UI label for a standard or custom field body or sublist field.
	Field.id	string (read-only)	Client and server-side scripts	Returns the internal ID of a standard or custom body or sublist field.
	Field.type	string (read-only)	Client and server-side scripts	Returns the type of a body or sublist field.
	Field.isMandatory	boolean <b>true</b>   <b>false</b>	Client and server-side scripts	Returns <b>true</b> if the standard or custom field is mandatory on the record form, or <b>false</b> otherwise.
	Field.sublistId	string (read-only)	Client and server-side scripts	Returns the ID of the sublist associated with the specified sublist field.
	Field.isDisplay	boolean <b>true</b>   <b>false</b>	Client and server-side scripts	Returns <b>true</b> if the field is visible on the record form, or <b>false</b> if it is not.

## 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](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Macro.execute(options)	Object	Client and server-side scripts	Performs a macro operation and returns its result in an object.
	Macro.execute.promise(options)	Promise	Client scripts	Performs a macro operation asynchronously.
	Macro(options)	Object	Client and server-side scripts	Performs a macro operation and returns its result in an object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	<a href="#">Macro.promise(options)</a>	Promise	Client scripts	Performs a macro operation asynchronously.
Property	<a href="#">Macro.id</a>	string	Client and server-side scripts	The ID of the macro. For a list of macro IDs, see the help topic <a href="#">Supported Record Macros</a>
	<a href="#">Macro.label</a>	string	Client and server-side scripts	The macro label.
	<a href="#">Macro.description</a>	string	Client and server-side scripts	The macro description.
	<a href="#">Macro.attributes</a>	Object	Client and server-side scripts	The macro defined attributes.

## Record Object Members

The following members are called on the [record.Record](#) object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">Record.cancelLine(options)</a>	<a href="#">record.Record</a>	Client and server-side scripts	Cancels the currently selected line on a sublist.
	<a href="#">Record.commitLine(options)</a>	<a href="#">record.Record</a>	Client and server-side scripts	Commits the currently selected line on a sublist.
	<a href="#">Record.executeMacro(options)</a>	Object	Client and server-side scripts	Performs macro operation and returns its result in a plain JavaScript object.
	<a href="#">Record.getMacros(options)</a>	Object	Client and server-side scripts	Provides a plain JavaScript object that contains macro objects defined for a record type, indexed by the Macro ID.
	<a href="#">Record.findMatrixSublistLineWithValue(options)</a>	number	Client and server-side scripts	Returns the line number of the first instance where a specified value is found in a specified column of the matrix.
	<a href="#">Record.findSublistLineWithValue(options)</a>	number	Client and server-side scripts	Returns the line number for the first occurrence of a field value in a sublist.
	<a href="#">Record.getCurrentMatrixSublistValue(options)</a>	number   Date   string   array   boolean <code>true</code>   <code>false</code>	Client and server-side scripts	Gets the value for the currently selected line in the matrix.
	<a href="#">Record.getCurrentSublistField(options)</a>	<a href="#">record.Field</a>	Client and server-side scripts	Returns a field object from a sublist.
	<a href="#">Record.getCurrentSublistIndex(options)</a>	number	Client and server-side scripts	Returns the line number of the currently selected line.
	<a href="#">Record.getCurrentSublistSubrecord(options)</a>	<a href="#">record.Record</a>	Client and server-side scripts	Gets the subrecord for the associated sublist field on the current line.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	Record.getCurrentSublistText(options)	string	Client and server-side scripts	Returns a text representation of the field value in the currently selected line.
	Record.getCurrentSublistValue(options)	number   Date   string   array   boolean true   false	Client and server-side scripts	Returns the value of a sublist field on the currently selected sublist line.
	Record.getField(options)	record.Field	Client and server-side scripts	Returns a field object from a record.
	Record.getFields()	string[]	Client and server-side scripts	Returns the body field names (internal ids) of all the fields in the record, including machine header field and matrix header fields.
	Record.getLineCount(options)	number	Client and server-side scripts	Returns the number of lines in a sublist.
	Record.getMacro(options)	record.Macro	Client and server-side scripts	Provides a macro to execute.
	Record.getMacros(options)	Object	Client and server-side scripts	Provides a plain JavaScript object that contains macro objects defined for a record type, indexed by the Macro ID.
	Record.getMatrixHeaderCount(options)	number	Client and server-side scripts	Returns the number of columns for the specified matrix.
	Record.getMatrixHeaderField(options)	record.Field	Client and server-side scripts	Gets the field for the specified header in the matrix.
	Record.getMatrixHeaderValue(options)	number   Date   string   array   boolean true   false	Client and server-side scripts	Gets the value for the associated header in the matrix.
	Record.getMatrixSublistField(options)	record.Field	Client and server-side scripts	Gets the field for the specified sublist in the matrix.
	Record.getMatrixSublistValue(options)	number   Date   string   array   boolean true   false	Client and server-side scripts	Gets the value for the associated field in the matrix.
	Record.getSublist(options)	record.Sublist	Client and server-side scripts	Returns the specified sublist.
	Record.getSublists()	string[]	Client and server-side scripts	Returns all the names of all the sublists.
	Record.getSublistField(options)	record.Field	Client and server-side scripts	Returns a field object from a sublist.
	Record.getSublistFields(options)	string[]	Client and server-side scripts	Returns all the field names in a sublist.
	Record.getSublistSubrecord(options)	record.Record	Client and server-side scripts	Gets the subrecord associated with a sublist field. (standard mode only)
	Record.getSublistText(options)	string	Client and server-side scripts	Returns the value of a sublist field in a text representation.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	Record.getSublistValue(options)	number   Date   string   array   boolean <b>true</b>   <b>false</b>	Client and server-side scripts	Returns the value of a sublist field.
	Record.getSubrecord(options)	record.Record	Client and server-side scripts	Gets the subrecord for the associated field.
	Record.getText(options)	string	Client and server-side scripts	Returns the text representation of a field value.
	Record.getValue(options)	number   Date   string   array   boolean <b>true</b>   <b>false</b>	Client and server-side scripts	Returns the value of a field.
	Record.hasCurrentSublistSubrecord(options)	boolean <b>true</b>   <b>false</b>	Client and server-side scripts	Returns a value indicating whether the associated sublist field has a subrecord on the current line.
	Record.hasSublistSubrecord(options)	boolean <b>true</b>   <b>false</b>	Client and server-side scripts	Returns a value indicating whether the associated sublist field contains a subrecord.
	Record.hasSubrecord(options)	boolean <b>true</b>   <b>false</b>	Client and server-side scripts	Returns a value indicating whether the field contains a subrecord.
	Record.insertLine(options)	record.Record	Client and server-side scripts	Inserts a sublist line.
	Record.removeCurrentSublistSubrecord(options)	record.Record	Client and server-side scripts	Removes the subrecord for the associated sublist field on the current line.
	Record.removeLine(options)	record.Record	Client and server-side scripts	Removes a sublist line.
	Record.removeSublistSubrecord(options)	record.Record	Client and server-side scripts	Removes the subrecord for the associated sublist field. (standard mode only)
	Record.removeSubrecord(options)	record.Record	Client and server-side scripts	Removes the subrecord for the associated field.
	Record.save(options)	number	Client and server-side scripts	Submits a new record or saves edits to an existing record.  This method is not available to subrecords.
	Record.save.promise(options)	number	Client scripts	Submits a new record asynchronously or saves edits to an existing record asynchronously.  This method is not available to subrecords.
	Record.selectLine(options)	record.Record	Client and server-side scripts	Selects an existing line in a sublist.
	Record.selectNewLine(options)	record.Record	Client and server-side scripts	Selects a new line at the end of a sublist.
	Record.setCurrentMatrixSublistValue(options)	record.Record	Client and server-side scripts	Sets the value for the line currently selected in the matrix.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	Record.setCurrentSublistText(options)	record.Record	Client and server-side scripts	Sets the value for the field in the currently selected line by a text representation.
	Record.setCurrentSublistValue(options)	record.Record	Client and server-side scripts	Sets the value for the field in the currently selected line.
	Record.setMatrixHeaderValue(options)	record.Record	Client and server-side scripts	Sets the value for the associated header in the matrix.
	Record.setMatrixSublistValue(options)	record.Record	Client and server-side scripts	Sets the value for the associated field in the matrix.
	Record.setSublistText(options)	record.Record	Client and server-side scripts	Sets the value of a sublist field by a text representation. (standard mode only)
	Record.setSublistValue(options)	record.Record	Client and server-side scripts	Sets the value of a sublist field. (standard mode only)
	Record.setText(options)	record.Record	Client and server-side scripts	Sets the value of the field by a text representation.
	Record.setValue(options)	record.Record	Client and server-side scripts	Sets the value of a field.
Property	Record.id	number (read-only)	Client and server-side scripts	The internal ID of a specific record. This property is not available to subrecords.
	Record.isDynamic	boolean (read-only)	Client and server-side scripts	Indicates whether the record is in dynamic or standard mode.
	Record.type	string (read-only)	Client and server-side scripts	The record type. This property is not available to subrecords.

## Sublist Object Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Sublist.getColumn(options)	record.Column	Client and server-side scripts	Returns a column in the sublist.
Property	Sublist.id	string (read-only)	Client and server-side scripts	Returns the internal ID of the sublist.
	Sublist.isChanged	boolean true   false (read-only)	Client and server-side scripts	Indicates whether the sublist has changed on the record form.
	Sublist.isDisplay	boolean true   false (read-only)	Client and server-side scripts	Indicates whether the sublist is displayed on the record form.
	Sublist.type	string (read-only)	Client and server-side scripts	Returns the sublist type.

## N/record Module Script Samples

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to use the record module to create and save a Contact record..

**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.

```
/*
 * @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();
});
```

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to create and save a contact record using promise methods.



**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. For more information on debugging SuiteScript client scripts, see the help topic [Debugging Client Scripts](#).

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

require(['N/record'], function(record) {
    function createAndSaveContactRecordWithPromise() {
        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();
    });
});
```



**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to access sublists and a subrecord from a record. This sample requires the Advanced Number Inventory Management feature.

```

/**
 * @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
        });
        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();
});

```

**Note:** For additional script samples that include subrecords, see the help topic [SuiteScript 2.0 Scripting Subrecords](#).

**Note:** This sample script uses the **require** function so that you can copy it into the SuiteScript Debugger and test it. You must use the **define** function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to access sublists and a subrecord from a record using promise methods. This sample requires the Advanced Number Inventory Management feature.

**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. For more information on debugging SuiteScript client scripts, see the help topic [Debugging Client Scripts](#).

```

/**
 * @NApiVersion 2.x
 */

require(['N/record'], function(record) {
    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'
        });
        var recordId = rec.save();
    }, function(err) {
        log.error('Unable to create purchase order!', err.name);
    });
}
createPurchaseOrder();
});

```

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample 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](#).

```

/**
 * @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)
{
    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 **defaultValue** 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.

Record	Initialization Type	Values
All SuiteScript-supported records that support form customization.	customform	<customformid>
Assembly Build	assemblyitem	<assemblyitemid>
Cash Refund	entity	<entityid>
Cash Sale	entity	<entityid>
Charge Rule	entity	<entityid>
Check	entity	<entityid>
Credit Memo	entity	<entityid>
Customer Payment	entity	<entityid>
Customer Refund	entity	<entityid>
Deposit	disablepaymentfilters	<disablepaymentfilters>
Estimate	entity	<entityid>
Expense Report	entity	<entityid>
Invoice	entity	<entityid>
Item Receipt	entity	<entityid>
Non-Inventory Part	subtype	sale   resale   purchase
Opportunity	entity	<entityid>
Other Charge Item	subtype	sale   resale   purchase
Purchase Order	entity	<entityid>
Return Authorization	entity	<entityid>
Sales Order	entity	<entityid>
Script Deployment	script	<scriptid>
Service	subtype	sale   resale   purchase
Subscription Change Order	entity	<scriptid>
Tax Group	nexuscountry	<countrycode> See <a href="#">Country Codes Used for Initialization Parameters</a> .
Tax Type	country	<countrycode> See <a href="#">Country Codes Used for Initialization Parameters</a> .
Topic	parenttopic	<parenttopicid>
Vendor Bill	entity	<entityid>

Record	Initialization Type	Values
Vendor Payment	entity	<entityid>
Work Order	assemblyitem	<assemblyitemid>

## 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:

```
record.create('taxgroup', {nexuscountry: 'AR'});
```

Country Code	Country Name
AD	Andorra
AE	United Arab Emirates
AF	Afghanistan
AG	Antigua and Barbuda
AI	Anguilla
AL	Albania
AM	Armenia
AO	Angola
AQ	Antarctica
AR	Argentina
AS	American Samoa
AT	Austria
AU	Australia
AW	Aruba
AX	Aland Islands
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BD	Bangladesh
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BH	Bahrain
BI	Burundi

<b>Country Code</b>	<b>Country Name</b>
BJ	Benin
BL	Saint Barthélemy
BM	Bermuda
BN	Brunei Darrussalam
BO	Bolivia
BQ	Bonaire, Saint Eustatius, and Saba
BR	Brazil
BS	Bahamas
BT	Bhutan
BV	Bouvet Island
BW	Botswana
BY	Belarus
BZ	Belize
CA	Canada
CC	Cocos (Keeling) Islands
CD	Congo, Democratic People's Republic
CF	Central African Republic
CG	Congo, Republic of
CH	Switzerland
CI	Cote d'Ivoire
CK	Cook Islands
CL	Chile
CM	Cameroon
CN	China
CO	Colombia
CR	Costa Rica
CU	Cuba
CV	Cape Verde
CW	Curacao
CX	Christmas Island
CY	Cyprus
CZ	Czech Republic

<b>Country Code</b>	<b>Country Name</b>
DE	Germany
DJ	Djibouti
DK	Denmark
DM	Dominica
DO	Dominican Republic
DZ	Algeria
EA	Ceuta and Melilla
EC	Ecuador
EE	Estonia
EG	Egypt
EH	Western Sahara
ER	Eritrea
ES	Spain
ET	Ethiopia
FI	Finland
FJ	Fiji
FK	Falkland Islands
FM	Micronesia, Federal State of
FO	Faroe Islands
FR	France
GA	Gabon
GB	United Kingdom
GD	Grenada
GE	Georgia
GF	French Guiana
GG	Guernsey
GH	Ghana
GI	Gibraltar
GL	Greenland
GM	Gambia
GN	Guinea
GP	Guadeloupe

<b>Country Code</b>	<b>Country Name</b>
GQ	Equatorial Guinea
GR	Greece
GS	South Georgia
GT	Guatemala
GU	Guam
GW	Guinea-Bissau
GY	Guyana
HK	Hong Kong
HM	Heard and McDonald Islands
HN	Honduras
HR	Croatia/Hrvatska
HT	Haiti
HU	Hungary
IC	Canary Islands
ID	Indonesia
IE	Ireland
IL	Israel
IM	Isle of Man
IN	India
IO	British Indian Ocean Territory
IQ	Iraq
IR	Iran (Islamic Republic of)
IS	Iceland
IT	Italy
JE	Jersey
JM	Jamaica
JO	Jordan
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KH	Cambodia
KI	Kiribati

<b>Country Code</b>	<b>Country Name</b>
KM	Comoros
KN	Saint Kitts and Nevis
KP	Korea, Democratic People's Republic
KR	Korea, Republic of
KW	Kuwait
KY	Cayman Islands
KZ	Kazakhstan
LA	Lao People's Democratic Republic
LB	Lebanon
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
LY	Libya
MA	Morocco
MC	Monaco
MD	Moldova, Republic of
ME	Montenegro
MF	Saint Martin
MG	Madagascar
MH	Marshall Islands
MK	Macedonia
ML	Mali
MM	Myanmar
MN	Mongolia
MO	Macau
MP	Northern Mariana Islands
MQ	Martinique

<b>Country Code</b>	<b>Country Name</b>
MR	Mauritania
MS	Montserrat
MT	Malta
MU	Mauritius
MV	Maldives
MW	Malawi
MX	Mexico
MY	Malaysia
MZ	Mozambique
NA	Namibia
NC	New Caledonia
NE	Niger
NF	Norfolk Island
NG	Nigeria
NI	Nicaragua
NL	Netherlands
NO	Norway
NP	Nepal
NR	Nauru
NU	Niue
NZ	New Zealand
OM	Oman
PA	Panama
PE	Peru
PF	French Polynesia
PG	Papua New Guinea
PH	Philippines
PK	Pakistan
PL	Poland
PM	St. Pierre and Miquelon
PN	Pitcairn Island
PR	Puerto Rico

<b>Country Code</b>	<b>Country Name</b>
PS	State of Palestine
PT	Portugal
PW	Palau
PY	Paraguay
QA	Qatar
RE	Reunion Island
RO	Romania
RS	Serbia
RU	Russian Federation
RW	Rwanda
SA	Saudi Arabia
SB	Solomon Islands
SC	Seychelles
SD	Sudan
SE	Sweden
SG	Singapore
SH	Saint Helena
SI	Slovenia
SJ	Svalbard and Jan Mayen Islands
SK	Slovak Republic
SL	Sierra Leone
SM	San Marino
SN	Senegal
SO	Somalia
SR	Suriname
SS	South Sudan
ST	Sao Tome and Principe
SV	El Salvador
SX	Sint Maarten
SY	Syrian Arab Republic
SZ	Swaziland
TC	Turks and Caicos Islands

<b>Country Code</b>	<b>Country Name</b>
TD	Chad
TF	French Southern Territories
TG	Togo
TH	Thailand
TJ	Tajikistan
TK	Tokelau
TM	Turkmenistan
TN	Tunisia
TO	Tonga
TP	East Timor
TR	Turkey
TT	Trinidad and Tobago
TV	Tuvalu
TW	Taiwan
TZ	Tanzania
UA	Ukraine
UG	Uganda
UM	US Minor Outlying Islands
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

Country Code	Country Name
YT	Mayotte
ZA	South Africa
ZM	Zambia
ZW	Zimbabwe

## record.Column



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a column of a sublist on a standard or custom record.  For a complete list of this object's properties, see <a href="#">Column Object Members</a> .  This object does not return a value, it returns information about the sublist column.
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	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](#).

```
// 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the internal ID of the column. Note that the Column.id value is the same as the value that is passed into fieldID.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Column Object Members</a>
<b>Since</b>	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](#).

```
// 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'
});
log.debug ({
    title: 'ID comparison',
    details: 'Note that objColumn.id = ' + objColumn.id + ' is the same as the value you passed in as fieldID.'
})...
// Add additional code.
```

## Column.label



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the internal ID of the column.  This property does not return a value, it returns information about the column label.
-----------------------------	---

<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	<a href="#">Column Object Members</a>
<b>Since</b>	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](#).

```
// 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
}
...
// Add additional code.
```

## Column.sublistId

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the internal ID of the standard or custom sublist that contains the column.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	<a href="#">Column Object Members</a>
<b>Since</b>	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](#).

```
// 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

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the column type. For more information on possible return values, see <a href="#">format.Type</a> .
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Column Object Members</a>
<b>Since</b>	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](#).

```
// 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.

```

## record.Field



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a body or sublist field on a standard or custom record.  Use the following methods to access the Field object: <ul style="list-style-type: none"><li>▪ <a href="#">Record.getField(options)</a></li><li>▪ <a href="#">Record.getSublistField(options)</a></li><li>▪ <a href="#">Record.getCurrentSublistField(options)</a></li><li>▪ <a href="#">CurrentRecord.getField(options)</a></li><li>▪ <a href="#">CurrentRecord.getSublistField(options)</a></li></ul> For a complete list of this object's methods and properties, see <a href="#">Field Object Members</a> .
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	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](#).

```

// 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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns an array of available options on a standard or custom select, multi-select, or radio field as key-value pairs.   <b>Important:</b> You can only use this method on a record in dynamic mode. For additional information on dynamic mode, see <a href="#">record.Record</a> and <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> .
<b>Returns</b>	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:  <div style="background-color: #f0f0f0; padding: 5px;"><code>[{value: 5, text: 'abc'},{value: 6, text: '123'}]</code></div> This function returns <b>Type Error</b> if the field is not a supported field for this method.
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Field Object Members</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.filter</code>	string	Required	The search string to filter the select options that are returned.  <div style="background-color: #e0f2ff; padding: 5px;"> <b>Note:</b> Filter values are case insensitive.</div>	2015.2
<code>options.operator</code>	string	Required	The following operators are supported:	2015.2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> <li>▪ <b>contains</b> (default)</li> <li>▪ <b>is</b></li> <li>▪ <b>startswith</b></li> </ul>	

## Syntax

**⚠ 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 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

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the UI label for a standard or custom field body or sublist field.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Field Object Members</a>
<b>Since</b>	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](#).

```
// 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
}
...
// Add additional code.

```

## Field.id



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the internal ID of a standard or custom body or sublist field.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Field Object Members</a>
<b>Since</b>	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](#).

```

// 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.type



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the type of a body or sublist field.  For example, the value can return <code>text</code> , <code>date</code> , <code>currency</code> , <code>select</code> , <code>checkbox</code> , etc. For more information on possible return values, see <a href="#">format.Type</a> .  The maximum character limit for select field types is 801.
<b>Type</b>	<code>string</code> (read-only)
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Field Object Members</a>
<b>Since</b>	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](#).

```
// 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.type === 'checkbox'){
  //Perform an action
}

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

## Field.isMandatory



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns <code>true</code> if the standard or custom field is mandatory on the record form, or <code>false</code> otherwise.
<b>Type</b>	<code>boolean true   false</code>
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	<a href="#">Field Object Members</a>
<b>Since</b>	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](#).

```
// 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.'
    };
    dialog.alert(options);
}
// Add additional code.
```

## Field.sublistId



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the sublist ID for the specified sublist field.
<b>Type</b>	<code>string</code> (read-only)

<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Field Object Members</a>
<b>Since</b>	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](#).

```
// 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

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns <b>true</b> if the field is visible on the record form, or <b>false</b> if it is not.
<b>Type</b>	boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Field Object Members</a>
<b>Since</b>	2015.2

## Syntax

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

```
define(['N/currentRecord'], function(currentRecord)
{return {pageInit: function(context) {
    var record = currentRecord.get();
    var field = record.getField({fieldId:"custbody1"});
    field.isDisplay = false;
    ...
// Add additional code.
```

## record.Macro



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a record macro. For information about record macros, see the help topic <a href="#">Overview of Record Action and Macro APIs</a> .  Use the <a href="#">Record.getMacro(options)</a> method to access the Macro object.  For a complete list of this object's methods and properties, see <a href="#">Macro Object Members</a> .
<b>Supported Script Types</b>	Client and server-side scripts.  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Methods and Properties</b>	<a href="#">Macro Object Members</a>
<b>Since</b>	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](#).

```
// Add additional code
...
var myMacro = record.getMacro({id: 'calculateTax'})
...
// Add additional code
```

## Macro.execute(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Performs a macro operation and returns its result in a plain JavaScript object.  For information about record macros, see the help topic <a href="#">Overview of Record Action and Macro APIs</a> .
<b>Returns</b>	{notifications: [], response: {}}
<b>Supported Script Types</b>	Client and server scripts

	For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Parent Object</b>	<a href="#">record.Macro</a>
<b>Sibling Object Members</b>	<a href="#">Macro Object Members</a>
<b>Since</b>	2018.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.params	Object	optional	The macro arguments.

## Syntax

 **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
...
timesheet.executeMacro({id:'copyFromWeek', params: {weekOf : '7/10/2017', copyExact : true}});
...
// Add additional code
```

## Macro.execute.promise(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Performs a macro operation asynchronously.  For information about record macros, see the help topic <a href="#">Overview of Record Action and Macro APIs</a> .   <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">Macro.execute(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Supported Script Types</b>	Client-side scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Parent Object</b>	<a href="#">record.Macro</a>
<b>Sibling Object Members</b>	<a href="#">Macro Object Members</a>
<b>Since</b>	2018.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.params	Object	optional	The macro arguments.

## Syntax

 **Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// Add additional code
...
myMacro.execute.promise().then(function(result){ /* do something with macro result */ });
...
// Add additional code
```

## Macro(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Performs a macro operation and returns its result in a plain JavaScript object.   <b>Note:</b> Substitute Macro with the name of the macro you are executing.
<b>Returns</b>	{notifications: [], response: {}}
<b>Supported Script Types</b>	Client and server scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Parent Object</b>	<a href="#">record.Macro</a>
<b>Sibling Object Members</b>	<a href="#">Macro Object Members</a>
<b>Since</b>	2018.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.params	Object	optional	The macro arguments.

## Syntax



**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 calculateTax = recordObj.getMacro({id: 'calculateTax'});
calculateTax();
...
// Add additional code
```

## Macro.promise(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Performs a macro operation asynchronously.
	<p><b>Note:</b> Substitute Macro with the name of the macro you are executing.</p> <p>For information about record macros, see the help topic <a href="#">Overview of Record Action and Macro APIs</a>.</p> <p><b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">Macro(options)</a>. For more information on promises, see <a href="#">Promise Object</a>.</p>
<b>Returns</b>	Promise Object
<b>Supported Script Types</b>	Client-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Parent Object</b>	<a href="#">record.Macro</a>
<b>Sibling Object Members</b>	<a href="#">Macro Object Members</a>
<b>Since</b>	2018.2

## Parameters



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

Parameter	Type	Required / Optional	Description
options.params	Object	optional	The macro arguments.

## Syntax



**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// Add additional code
```

```

...
var calculateTax = recordObj.getMacro({id: 'calculateTax'});
calculateTax.promise();
...
// Add additional code

```

## Macro.id



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The ID of the macro.  For information about record macros, see the help topic <a href="#">Overview of Record Action and Macro APIs</a> .
<b>Type</b>	string
<b>Supported Script Types</b>	Client and server scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Parent Object</b>	<a href="#">record.Macro</a>
<b>Sibling Object Members</b>	<a href="#">Macro Object Members</a>
<b>Since</b>	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](#).

```

// Add additional code
...
var id = macro.id; // get the id of the macro
...
// Add additional code

```

## Macro.label



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The label of the macro.  For information about record macros, see the help topic <a href="#">Overview of Record Action and Macro APIs</a> .
<b>Type</b>	string
<b>Supported Script Types</b>	Client and server scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Module</b>	N/record Module
<b>Parent Object</b>	record.Macro
<b>Sibling Object Members</b>	Macro Object Members
<b>Since</b>	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](#).

```
// Add additional code
...
var label = macro.label; // get the label of the macro
...
// Add additional code
```

## Macro.description



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The description of the macro.  For information about record macros, see the help topic <a href="#">Overview of Record Action and Macro APIs</a> .
<b>Type</b>	string
<b>Supported Script Types</b>	Client and server scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/record Module
<b>Parent Object</b>	record.Macro
<b>Sibling Object Members</b>	Macro Object Members
<b>Since</b>	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](#).

```
// Add additional code
...
var description = macro.description; // get the description of the macro
...
// Add additional code
```

## Macro.attributes



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The defined attributes of the macro.  For information about record macros, see the help topic <a href="#">Overview of Record Action and Macro APIs</a> .
<b>Type</b>	Object
<b>Supported Script Types</b>	Client and server scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Parent Object</b>	<a href="#">record.Macro</a>
<b>Sibling Object Members</b>	<a href="#">Macro Object Members</a>
<b>Since</b>	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](#).

```
// Add additional code
...
var attributes = macro.attributes; // get the attributes of the macro
...
// Add additional code
```

## record.Record



**Note:** The content in this help topic pertains to SuiteScript 2.0.

Load the record module when you want to work with NetSuite records.

<b>Object Description</b>	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.  <ul style="list-style-type: none"> <li>■ 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 <a href="#">Record.save(options)</a>.  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.</li> <li>■ 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.</li> </ul>
---------------------------	--

	<p>The <a href="#">record.create(options)</a>, <a href="#">record.copy(options)</a>, <a href="#">record.load(options)</a>, and <a href="#">record.transform(options)</a> 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.</p> <p>Use <a href="#">record.Type</a> enum for multiple records. For help finding a record's internal ID, see the help topic <a href="#">How do I find a record's internal ID?</a></p> <p>For more information about standard and dynamic modes, see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a></p> <p>For a complete list of this object's methods and properties, see <a href="#">Record Object Members</a>.</p>
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	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](#).

```
// Add additional code.

...
var objRecord = record.load({
    type: record.Type.SALES_ORDER,
    id: '6',
    isDynamic: true
});
...
// Add additional code.
```

## Record.cancelLine(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Cancels the currently selected line on a sublist. (dynamic mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	The <a href="#">record.Record</a> object that called the method.
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax

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

## Record.commitLine(options)

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Commits the currently selected line on a sublist. (dynamic mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )  When working in standard mode, set a sublist field using <a href="#">Record.setSublistValue(options)</a> .
<b>Returns</b>	The <a href="#">record.Record</a> object that called the method.
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>

<b>Since</b>	2015.2
--------------	--------

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.ignoreRecalc	boolean true   false	optional	If set to true, scripting recalculation is ignored.	2015.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax

<b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/record Module Script Samples</a> .
--

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

## Record.executeMacro(options)

<b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.
--

<b>Method Description</b>	Performs macro operation and returns its result in a plain JavaScript object.  For information about record macros, see the help topic <a href="#">Overview of Record Action and Macro APIs</a> .
<b>Returns</b>	An object with the macro results or <code>null</code> .
<b>Supported Script Types</b>	Client and server scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	Record Object Members
<b>Since</b>	2018.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.id	string	required	The macro ID.
options.params	Object	optional	The macro arguments.

## Syntax

 **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
...
if ('calculateTax' in macros) {
    macros.calculateTax();
}
...
// Add additional code
```

## Record.executeMacro.promise(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Performs macro operation and returns its result in a plain JavaScript object.  For information about record macros, see the help topic <a href="#">Overview of Record Action and Macro APIs</a> .
<b>Note:</b>	The parameters and errors thrown for this method are the same as those for <a href="#">Record.executeMacro(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Supported Script Types</b>	Client-side scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	Record Object Members
<b>Since</b>	2018.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.id	string	required	The macro ID.
options.params	Object	optional	The macro arguments.

## Syntax

**⚠ Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// Add additional code
...
if ('calculateTax' in macros) {
    macros.calculateTax.promise();
}
...
// Add additional code
```

## Record.findMatrixSublistLineWithValue(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	number
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.	2016.2

Parameter	Type	Required / Optional	Description	Since
			This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	
<code>options.fieldId</code>	string	required	The ID of the matrix field. See, <a href="#">How do I find a field's internal ID?</a>	2016.2
<code>options.value</code>	number	required	The value to search for.	2016.2
<code>options.column</code>	number	required	The column number of the field. Note that column indexing begins at 0 with SuiteScript 2.0.	2016.2

## Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	A required argument is missing or undefined.
<code>SSS_INVALID_SUBLIST_OPERATION</code>	A required argument is invalid or the sublist is not editable.

## Syntax

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

## Record.findSublistLineWithValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.  (dynamic and standard mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	A line number as a number, or -1 if not found.
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>

<b>Sibling Object Members</b>	Record Object Members
<b>Since</b>	2015.2

## Parameters

<b>i</b> <b>Note:</b> The options parameter is a JavaScript object.
---

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
options.value	number   Date   string   array   boolean true   false	optional	The value to search for.	2015.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or not defined.

## 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](#).

```
// Add additional code.

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

## Record.getCurrentMatrixSublistValue(options)

<b>i</b> <b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.
---

<b>Method Description</b>	Gets the value for the currently selected line in the matrix.
---------------------------	---

	(dynamic mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> ) Gets a numeric value for rate and ratehighprecision fields.
<b>Returns</b>	number   Date   string   array   boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	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 <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of the matrix field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
options.column	number	required	The column number for the matrix field. Note that column indexing begins at 0 with SuiteScript 2.0.	2015.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

## Syntax

 <b>Important:</b>	The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/record Module Script Samples</a> .
---	--

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

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

## Record.getCurrentSublistField(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns metadata about a sublist field.  (dynamic mode only— see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	<a href="#">record.Field</a>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2016.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2

### Errors

Error Code	Thrown If
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 script example, see [N/record Module Script Samples](#).

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

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

## Record.getCurrentSublistIndex(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	number
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2

### Errors

Error Code	Thrown If
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 script example, see [N/record Module Script Samples](#).

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

```

...
var curriIndex = objRecord.getCurrentSublistIndex({
    sublistId: 'item'
});
...
// Add additional code.

```

## Record.getCurrentSublistSubrecord(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the subrecord for the associated sublist field on the current line.  (dynamic mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	<a href="#">record.Record</a>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See, <a href="#">How do I find a field's internal ID?</a>	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](#).

```

// Add additional code.
...
var objSubrecord = objRecord.getCurrentSublistSubrecord({
    sublistId: 'item',

```

```

    fieldId: 'item'
});
...
// Add additional code.

```

## Record.getCurrentSublistText(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a text representation of the field value in the currently selected line. (dynamic mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> ) Gets a string value with a "%" for rate and ratehighprecision fields.
<b>Returns</b>	string   <b>Note:</b> For multiselect fields, returns an array.
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2

### Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax

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

## Record.getCurrentSublistValue(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the value of a sublist field on the currently selected sublist line. (dynamic mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	number   Date   string   array   boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax

 **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 sublistValue = objRecord.getCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'item'
});
...
// Add additional code.
```

## Record.getField(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a field object from a record.  (dynamic and standard modes — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	<code>record.Field</code>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.fieldId</code>	string	required	The internal ID of a standard or custom body field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2

## Errors

Error Code	Thrown If
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 script example, see [N/record Module Script Samples](#).

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

## Record.getFields()

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	string[]
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	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](#).

```
// Add additional code.
...
var objFields = objRecord.getFields();
...
// Add additional code.
```

## Record.getLineCount(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the number of lines in a sublist.  (standard and dynamic mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	number
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	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](#).

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

## Record.getMacro(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Provides a macro to be executed.
---------------------------	----------------------------------

	For information about record macros, see the help topic <a href="#">Overview of Record Action and Macro APIs</a> .
<b>Returns</b>	Function to be executed for the macro.
<b>Supported Script Types</b>	Client and server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2018.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.id	string	required	The macro ID.

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required parameter is missing.
SSS_INVALID_MACRO_ID	A macro does not exist on the record.

## Syntax



**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 macro = recordObj.getMacro({id: 'calculateTax'});
...
// Add additional code
```

## Record.getMacros(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">record.Macro</a> objects. If there are no macros available for the specified record type, an empty object is returned.
---------------------------	---

	For information about record macros, see the help topic <a href="#">Overview of Record Action and Macro APIs</a> .
<b>Returns</b>	Object
<b>Supported Script Types</b>	Client and server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2018.2

## Errors

Error Code	Thrown If
SSS_INVALID_RECORD_TYPE	The specified record type is invalid.

## Syntax



**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 macroList = recordObj.getMacros();
...
// Add additional code
```

## Record.getMatrixHeaderCount(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the number of columns for the specified matrix.  (standard and dynamic mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	number
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	<p>The internal ID of the sublist that contains the matrix.</p> <p>This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a>.</p>	2015.2
options.fieldId	string	required	<p>The internal ID of the matrix field.</p> <p>See, <a href="#">How do I find a field's internal ID?</a></p>	2015.2

## Errors

Error Code	Thrown If
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 script example, see [N/record Module Script Samples](#).

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

## Record.getMatrixHeaderField(options)

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Gets the field for the specified header in the matrix.  (standard and dynamic mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	<a href="#">record.Field</a>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/record Module

<b>Sibling Object Members</b>	Record Object Members
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	<p>The internal ID of the sublist that contains the matrix.</p> <p>This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a>.</p>	2015.2
options.fieldId	string	required	<p>The internal ID of the matrix field.</p> <p>See, <a href="#">How do I find a field's internal ID?</a></p>	2015.2
options.column	number	required	<p>The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.</p>	2015.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

## Syntax

 <b>Important:</b>	The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/record Module Script Samples</a> .
---	--

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

## Record.getMatrixHeaderValue(options)

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Gets the value for the associated header in the matrix.  (standard and dynamic mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
---------------------------	---

	Gets a numeric value for rate and ratehighprecision fields.
<b>Returns</b>	number   Date   string   array   boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	Record Object Members
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	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 <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of the matrix field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
options.column	number	required	The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.	2015.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

## Syntax

 <b>Important:</b>	The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/record Module Script Samples</a> .
---	--

```
// Add additional code.

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

## Record.getMatrixSublistField(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the field for the specified sublist in the matrix.  (standard and dynamic mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	<code>record.Field</code>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
<code>options.sublistId</code>	string	required	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 <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
<code>options.fieldId</code>	string	required	The internal ID of the matrix field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
<code>options.column</code>	number	required	The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.	2015.2
<code>options.line</code>	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	2015.2

### Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	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](#).

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

```

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

```

## Record.getMatrixSublistValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the value for the associated field in the matrix.  (standard and dynamic mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	number   Date   string   array   boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	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 <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of the matrix field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
options.column	number	required	The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.	2015.2
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	2015.2

## Errors

Error Code	Thrown If
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 script example, see [N/record Module Script Samples](#).

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

## Record.getSublist(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the specified sublist.  (standard and dynamic mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	record.Sublist
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	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](#).

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

## Record.getSublists()

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns all the names of all the sublists.  (standard and dynamic mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	string[]
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	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](#).

```
// Add additional code.  
...  
var sublistName = objRecord.getSublists();  
...  
// Add additional code.
```

## Record.getSublistField(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a field object from a sublist.
---------------------------	--

	(standard and dynamic mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	<code>record.Field</code>
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.sublistId</code>	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
<code>options.fieldId</code>	string	required	The internal ID of a standard or custom sublist field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
<code>options.line</code>	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	2015.2

## Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	A required argument is missing or undefined.
<code>SSS_INVALID_SUBLIST_OPERATION</code>	A required argument is invalid or the sublist is not editable.

## Syntax

 **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 objField = objRecord.getSublistField({
    sublistId: 'item',
    fieldId: 'item',
    line: 3
```

```
});  
...  
// Add additional code.
```

## Record.getSublistFields(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns all the field names in a sublist.  (standard and dynamic mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	string[]
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2

### Errors

Error Code	Thrown If
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 script example, see [N/record Module Script Samples](#). For other samples, see [Sublist.getColumn\(options\)](#).

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

```

...
var field = objRecord.getSublistFields({
    sublistId: 'item'
});
...
// Add additional code.

```

## Record.getSublistSubrecord(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the subrecord associated with a sublist field. (standard mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )  When working in dynamic mode, get a sublist subrecord using the following methods: <ol style="list-style-type: none"> <li>1. <a href="#">Record.selectLine(options)</a></li> <li>2. <a href="#">Record.hasCurrentSublistSubrecord(options)</a></li> <li>3. <a href="#">Record.getCurrentSublistSubrecord(options)</a></li> </ol>
<b>Returns</b>	<a href="#">record.Record</a>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
options.line	number	required	The line number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.	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](#).

```
// Add additional code.

...
var objSubRecord = objRecord.getSublistSubrecord({
    sublistId: 'item',
    fieldId: 'item',
    line: 3
});
...
// Add additional code.
```

## Record.getSublistText(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the value of a sublist field in a text representation.  (standard mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )  Gets a string value with a "%" for rate and ratehighprecision fields.
<b>Returns</b>	string  <b>Note:</b> For multiselect fields, returns an array.
<b>Supported Script Types</b>	Client and server-side scripts.  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .  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 <a href="#">Errors</a> table.  In dynamic mode, you can use getSublistText() without limitation.
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.	2015.2

Parameter	Type	Required / Optional	Description	Since
			This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	
<code>options.fieldId</code>	string	required	The internal ID of a standard or custom sublist field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
<code>options.line</code>	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	2015.2

## Errors

Error Code	Thrown If
SSS_INVALID_API_USAGE	<p>Invoked in certain cases when deferredDynamic mode is being used. For example, if <code>Record.isDynamic</code> is set to false, this error can be invoked in both of the following situations:</p> <ul style="list-style-type: none"> <li>■ If the record object was created by <code>record.copy()</code>, <code>record.create()</code>, or <code>record.transform()</code>, and the script attempts to use <code>getSublistText()</code> without first using <code>setSublistText()</code> for the same field.</li> <li>■ If the record object was created by <code>record.load()</code>, and the script uses <code>setSublistValue()</code> on a field before using <code>getSublistText()</code> for the same field.</li> </ul> <p>This guidance also affects user event scripts that instantiate records by using the <code>newRecord</code> or <code>oldRecord</code> object provided by the script context. These records always use deferredDynamic mode. For that reason, this error appears in both of the following situations:</p> <ul style="list-style-type: none"> <li>■ When a user event script executes on a record that is being newly created, and the script attempts to use <code>getSublistText()</code> without first using <code>setSublistText()</code> for the same field.</li> <li>■ When a user event script executes on an existing record, and the script uses <code>setSublistValue()</code> on a field before using <code>getSublistText()</code> for the same field.</li> </ul> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a>.</p>
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax



**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 sublistFieldName = objRecord.getSublistText({
    sublistId: 'item',
    fieldId: 'item',
    line: 3
})
```

```
});  
...  
// Add additional code.
```

## Record.getSublistValue(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the value of a sublist field.  (dynamic and standard modes — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )  Gets a numeric value for rate and ratehighprecision fields.
<b>Returns</b>	number   Date   string   array   boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	2015.2

### Errors

Error Code	Thrown If
SSS_INVALID_API_USAGE	Invoked prior to using setSublistValue in standard record mode.
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax

**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 sublistFieldValue = objRecord.getSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    line: 3
});
...
// Add additional code.
```

## Record.getSubrecord(options)

<b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.	
<b>Method Description</b>	Gets the subrecord for the associated field.  This method is not available for subrecords.  (dynamic and standard mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	<a href="#">record.Record</a>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.				
<b>Parameter</b>	<b>Type</b>	<b>Required / Optional</b>	<b>Description</b>	<b>Since</b>
options.fieldId	string	required	The internal ID of a standard or custom body field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

Error Code	Thrown If
FIELD_1_IS_NOT_A_SUBRECORD_FIELD	The specified field is not a subrecord field.
FIELD_1_IS_DISABLED_YOU_CANNOT_APPLY_SUBRECORD_OPERATION_ON_THIS_FIELD	The specified field is disabled.

## Syntax

 **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 sublistFieldValue = objRecord.getSubrecord({
    fieldId: 'idnumber'
});
...
// Add additional code.
```

## Record.getText(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the text representation of a field value.  (dynamic and standard mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )  Gets a string value with a "%" for rate and ratehighprecision fields.
<b>Returns</b>	string   <b>Note:</b> For multiselect fields, returns an array.
<b>Supported Script Types</b>	Client and server-side scripts.  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .  In dynamic mode, you can use <code>getText()</code> without limitation but, in standard mode, limitations exist. In standard mode, you can use this method <b>only</b> in the following cases: <ul style="list-style-type: none"> <li>■ You can use <code>getText()</code> on any field where the script has already used <code>setText()</code>.</li> <li>■ If you are loading or copying a record, you can use <code>getText</code> on any field except those where the script has already changed the value by using <code>setValue()</code>.</li> </ul> For more details, refer to the description of the <code>SSS_INVALID_API_USAGE</code> error code in the <a href="#">Errors</a> table.
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	The internal ID of a standard or custom body field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_API_USAGE	Invoked in certain cases when standard mode is being used.  For example, if <code>Record.isDynamic</code> is set to false, the <code>SSS_INVALID_API_USAGE</code> error can be invoked in the following situations: <ul style="list-style-type: none"><li>■ 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.</li><li>■ 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.</li></ul> 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 <code>SSS_INVALID_API_USAGE</code> error appears when a user event executes on one of these objects in the following situations: <ul style="list-style-type: none"><li>■ 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.</li><li>■ 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.</li></ul>

## Syntax

 <b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/record Module Script Samples</a> .
--

```
// Syntax Sample 1
// Add additional code.
...
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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the value of a field.  (dynamic and standard mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )  Gets a numeric value for rate and ratehighprecision fields.
<b>Returns</b>	number   Date   string   array   boolean <b>true</b>   <b>false</b>  Returns a JavaScript Date object for date/time field queries. To return a string for date/time field queries, use <a href="#">Record.getText(options)</a> . Date/time fields: DATE, DATETIME, DATETIMETZ, TIMEOFTDAY.  <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <span style="color: #0070C0; font-size: 1.5em;">i</span> <b>Note:</b> If the returned date object is implicitly converted to a string, the value is converted using the browser's setting for time zone.         </div>
	All fields of type CHECKBOX return <b>true</b> or <b>false</b> .
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	The internal ID of a standard or custom body field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2

### Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

Error Code	Thrown If
SSS_INVALID_API_USAGE	Invoked in standard mode, if you use setText on a field and then use getValue on the same 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/record Module Script Samples](#).

```
// Add additional code.  
...  
var value = objRecord.getValue({  
    fieldId: 'item'  
});  
...  
// Add additional code.
```

## Record.hasCurrentSublistSubrecord(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a value indicating whether the associated sublist field has a subrecord on the current line.  (dynamic mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of a subrecord.  See, <a href="#">How do I find a field's internal ID?</a>	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](#).

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

## Record.hasSublistSubrecord(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	boolean <code>true</code>   <code>false</code>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters



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

Parameter	Type	Required / Optional	Description	Since
<code>options.sublistId</code>	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
<code>options.fieldId</code>	string	required	The internal ID of a subrecord.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2

Parameter	Type	Required / Optional	Description	Since
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	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](#).

```
// Add additional code.
...
var hasSubrecord = objRecord.hasSublistSubrecord({
    sublistId: 'item',
    fieldId: 'item',
    line: 3
});
...
// Add additional code.
```

## Record.hasSubrecord(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	boolean <code>true</code>   <code>false</code>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	The internal ID of the field that may contain a subrecord.  See, <a href="#">How do I find a field's internal ID?</a>	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](#).

```
// Add additional code.  
...  
var hasSubrecord = objRecord.hasSubrecord({  
    fieldId: 'item'  
});  
...  
// Add additional code.
```

## Record.insertLine(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	<a href="#">record.Record</a>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.sublistId</code>	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
<code>options.line</code>	number	required	The line number to insert. Note that line indexing begins at 0 with SuiteScript 2.0.	2015.2

Parameter	Type	Required / Optional	Description	Since
options. ignoreRecalc	boolean <code>true</code>   <code>false</code>	optional	If set to true, scripting recalculation is ignored.  The default value is false.	2015.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## 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 the help topic [Example: Creating a Landed Cost Sublist Subrecord](#).

```
// 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 record Module Script Samples.

## Record.removeCurrentSublistSubrecord(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	<code>record.Record</code>

<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	Record Object Members
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See, <a href="#">How do I find a field's internal ID?</a>	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](#).

```
// Add additional code.
...
objRecord.removeCurrentSublistSubrecord({
    sublistId: 'item',
    fieldId: 'item'
});
...
// Add additional code.
```

## Record.removeLine(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Removes a sublist line.  (dynamic and standard mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	<code>record.Record</code>

<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	Record Object Members
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.line	number	required	The line number of the sublist to remove. Note that line indexing begins at 0 with SuiteScript 2.0.	2015.2
options.ignoreRecalc	boolean <code>true</code>   <code>false</code>	optional	If set to true, scripting recalculation is ignored.  The default value is false.	2015.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax

 <b>Important:</b>	The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/record Module Script Samples</a> .
---	--

```
// Add additional code.
...
objRecord.removeLine({
    sublistId: 'item',
    line: 3,
    ignoreRecalc: true
});
...
// Add additional code.
```

## Record.removeSublistSubrecord(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Removes the subrecord for the associated sublist field. (standard mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )  When working in dynamic mode, remove a sublist subrecord using the following methods: <ol style="list-style-type: none"><li>1. Record.selectLine(options)</li><li>2. Record.hasCurrentSublistSubrecord(options)</li><li>3. Record.removeCurrentSublistSubrecord(options)</li><li>4. Record.commitLine(options)</li></ol>
<b>Returns</b>	record.Record
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
options.line	number	required	The line number in the sublist that contains the subrecord to remove. Note that line indexing begins at 0 with SuiteScript 2.0.	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](#).

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

```

```

objRecord.removeSublistSubrecord({
    sublistId: 'item',
    fieldid: 'item',
    line: 3
});
...
// Add additional code.

```

## Record.removeSubrecord(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Removes the subrecord for the associated field.  This method is not available for subrecords.  (dynamic and standard mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	<a href="#">record.Record</a>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	The internal ID of a standard or custom body field.  See, <a href="#">How do I find a field's internal ID?</a>	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](#).

```

// Add additional code.
...
objRecord.removeSubrecord({
    fieldid: 'item'
});
...
// Add additional code.

```

## Record.save(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Submits a new record or saves edits to an existing record.</p> <p>When working with records in standard mode, you must submit and then load the record to obtain sourced, validated, and calculated field values.</p> <p>This method is not available to subrecords.</p> <p><b>Note:</b> This method has an asynchronous counterpart you can use with client scripts. See <a href="#">Record.save.promise(options)</a>.</p>
<b>Returns</b>	A number representing the internal ID of the new or updated record.
<b>Supported Script Types</b>	<p>Client and server-side scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Governance</b>	<p>Transaction records: 20 usage units</p> <p>Custom records: 4 usage units</p> <p>All other records: 10 usage units</p>
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.enableSourcing	boolean <code>true</code>   <code>false</code>	optional	<p>Enables sourcing during the record update.</p> <p>If set to <code>true</code>, sources dependent field information for empty fields.</p> <p>Defaults to <code>false</code> – dependent field values are not sourced.</p> <p><b>Important:</b> 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 <code>enableSourcing</code> is ignored. See the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a>.</p>	2015.2
options.ignoreMandatoryFields	boolean <code>true</code>   <code>false</code>	optional	Disables mandatory field validation for this save operation.	2015.2

Parameter	Type	Required / Optional	Description	Since
			<p>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.</p> <p>By default, this parameter is <code>false</code>.</p> <div style="border: 1px solid #f0e68c; padding: 10px; background-color: #fff;"> <span style="color: #f0adbe; font-size: 1.2em; margin-right: 5px;">!</span> <b>Important:</b> Use the <code>ignoreMandatoryFields</code> 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.         </div>	

## Syntax

! **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 recordId = objRecord.save({
    enableSourcing: true,
    ignoreMandatoryFields: true
});
...
// Add additional code.
```

## Record.save.promise(options)

i **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Submits a new record asynchronously or saves edits to an existing record asynchronously.  This method is not available to subrecords.
	<span style="color: #0070C0; font-size: 1.2em; margin-right: 5px;">i</span> <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">Record.save(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Supported Script Types</b>	Client-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	Transaction records: 20 usage units  Custom records: 4 usage units  All other records: 10 usage units
<b>Module</b>	<a href="#">N/record Module</a>

<b>Sibling Object Members</b>	Record Object Members
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
<code>options.enableSourcing</code>	boolean <code>true</code>   <code>false</code>	optional	<p>Enables sourcing during the record update.</p> <p>If set to <code>true</code>, sources dependent field information for empty fields.</p> <p>Defaults to <code>false</code> – dependent field values are not sourced.</p> <div style="border: 1px solid #f0e68c; padding: 5px; margin-top: 10px;">  <b>Important:</b> 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 <code>enableSourcing</code> is ignored. See the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a>.         </div>	2015.2
<code>options.ignoreMandatoryFields</code>	boolean <code>true</code>   <code>false</code>	optional	<p>Disables mandatory field validation for this save operation.</p> <p>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.</p> <p>By default, this parameter is <code>false</code>.</p> <div style="border: 1px solid #f0e68c; padding: 5px; margin-top: 10px;">  <b>Important:</b> Use the <code>ignoreMandatoryFields</code> 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.         </div>	2015.2

## Syntax

 <b>Important:</b> The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see <a href="#">Promise Object</a> .
---

```
// Add additional code.
...
var recordId = objRecord.save.promise({
    enableSourcing: true,
    ignoreMandatoryFields: true
});
...
// Add additional code.
```

## Record.selectLine(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Selects an existing line in a sublist. (dynamic mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )  When working in standard mode, set a sublist field using <a href="#">Record.setSublistValue(options)</a> .
<b>Returns</b>	<a href="#">record.Record</a>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.line	number	required	The line number to select in the sublist. Note that line indexing begins at 0 with SuiteScript 2.0.	2015.2

### Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

### Syntax

 **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 lineNumber = objRecord.selectLine({  
    sublistId: 'item',
```

```

    line: 3
});
...
// Add additional code.

```

## Record.selectNewLine(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Selects a new line at the end of a sublist.  (dynamic mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	<code>record.Record</code>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.sublistId</code>	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2

### Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	A required argument is missing or undefined.
<code>SSS_INVALID_SUBLIST_OPERATION</code>	A required argument is invalid or the sublist is not editable.

### Syntax

 **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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value for the line currently selected in the matrix.  (dynamic mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )  Sets a string value with a "%" for rate and ratehighprecision fields.
<b>Returns</b>	<code>record.Record</code>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
<code>options.sublistId</code>	string	required	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 <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
<code>options.fieldId</code>	string	required	The internal ID of the matrix field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
<code>options.column</code>	number	required	The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.	2015.2
<code>options.value</code>	number   Date   string   array   boolean <code>true</code>   <code>false</code>	required	The value to set the field to.  The value type must correspond to the field type being set. For example: <ul style="list-style-type: none"><li>■ Text, Radio and Select fields accept string values.</li></ul>	2015.2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> <li>■ Checkbox fields accept Boolean values.</li> <li>■ Date and DateTime fields accept Date values.</li> <li>■ Integer, Float, Currency and Percent fields accept number values.</li> </ul>	
options.ignoreFieldChange	boolean true   false	optional	If set to <b>true</b> , the field change and slaving event is ignored.  By default, this value is <b>false</b> .	2015.2

## Errors

Error Code	Thrown If
INVALID_FLD_VALUE	The options.value type does not match the field type.
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 script example, see [N/record Module Script Samples](#).

```
// Add additional code.
...
objRecord.setCurrentMatrixSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 3,
    value: false,
    ignoreFieldChange: true,
    forceSyncSourcing: true
});
...
// Add additional code.
```

## Record.setCurrentSublistText(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value for the field in the currently selected line by a text representation.  (dynamic mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	<a href="#">record.Record</a>
<b>Supported Script Types</b>	Client and server-side scripts

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
options.text	string	required	The text to set the value to.	2015.2
options.ignoreFieldChange	boolean <code>true</code>   <code>false</code>	optional	If set to <code>true</code> , the field change and slaving event is ignored.  By default, this value is <code>false</code> .	2015.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
A_SCRIPT_IS_ATTEMPTING_TO_EDIT_THE_1_SUBLIST_THIS_SUBLIST_IS_CURRENTLY_IN_READONLY_MODE_AND_CANNOT_BE_EDITED_CALL_YOUR_NETSUITE_ADMINISTRATOR_TO_DISABLE_THIS_SCRIPT_IF_YOU_NEED_TO_SUBMIT_THIS_RECORD	A user tries to edit a read-only sublist field.

## Syntax

 <b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/record Module Script Samples</a> .
--

```
// Add additional code.
...
objRecord.setCurrentSublistText({
    sublistId: 'item',
    fieldId: 'item',
    text: 'value',
    ignoreFieldChange: true
});
```

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

## Record.setCurrentSublistValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Sets the value for the field in the currently selected line. (dynamic mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a>)</p> <p>When working in standard mode, set a sublist field using <a href="#">Record.setSublistValue(options)</a>.</p> <p> <b>Important:</b> 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.</p> <p>Sets a numeric value for rate and ratehighprecision fields.</p>
<b>Returns</b>	<a href="#">record.Record</a>
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
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: <ul style="list-style-type: none"> <li>■ Text, Radio and Select fields accept string values.</li> </ul>	2015.2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> <li>■ Checkbox fields accept Boolean values.</li> <li>■ Date and DateTime fields accept Date values.</li> <li>■ Integer, Float, Currency and Percent fields accept number values.</li> </ul>	
options.ignoreFieldChange	boolean true   false	optional	If set to <b>true</b> , the field change and slaving event is ignored.  By default, this value is <b>false</b> .	2015.2

## Errors

Error Code	Thrown If
INVALID_FLD_VALUE	The options.value type does not match the field type.
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
A_SCRIPT_IS_ATTEMPTING_TO_EDIT_THE_1 sublist THIS sublist IS CURRENTLY IN READONLY_MODE_AND_CANNOT_BE_EDITED_CALL_YOUR_NETSUITE_ADMINISTRATOR_TO_DISABLE_THIS_SCRIPT_IF_YOU_NEED_TO_SUBMIT_THIS_RECORD	A user tries to edit a read-only sublist 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/record Module Script Samples](#).

```
// Add additional code.
...
objRecord.setCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    value: true,
    ignoreFieldChange: true
});
...
// Add additional code.
```

## Record.setMatrixHeaderValue(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value for the associated header in the matrix.  (dynamic and standard modes — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )  Sets a numeric value for rate and ratehighprecision fields.
<b>Returns</b>	<a href="#">record.Record</a>

<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	Record Object Members
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	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 <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of the matrix field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
options.column	number	required	The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.	2015.2
options.value	number   Date   string   array   boolean <b>true</b>   <b>false</b>	required	The value to set the field to.  The value type must correspond to the field type being set. For example: <ul style="list-style-type: none"><li>■ Text, Radio and Select fields accept string values.</li><li>■ Checkbox fields accept Boolean values.</li><li>■ Date and DateTime fields accept Date values.</li><li>■ Integer, Float, Currency and Percent fields accept number values.</li></ul>	2015.2
options.ignoreFieldChange	boolean <b>true</b>   <b>false</b>	optional	If set to <b>true</b> , the field change and slaving event is ignored.  By default, this value is <b>false</b> .	2015.2

## Errors

Error Code	Thrown If
INVALID_FLD_VALUE	The options.value type does not match the field type.

Error Code	Thrown If
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 script example, see [N/record Module Script Samples](#).

```
// Add additional code.
...
objRecord.setMatrixHeaderValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 3,
    value: false,
    ignoreFieldChange: true,
    forceSyncSourcing: true
});
...
// Add additional code.
```

## Record.setMatrixSublistValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value for the associated field in the matrix.  (standard mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )  Sets a numeric value for rate and ratehighprecision fields.
<b>Returns</b>	<code>record.Record</code>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.sublistId</code>	string	required	The internal ID of the sublist that contains the matrix.	2015.2

Parameter	Type	Required / Optional	Description	Since
			This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	
options.fieldId	string	required	The internal ID of the matrix field. See, <a href="#">How do I find a field's internal ID?</a>	2015.2
options.column	number	required	The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.	2015.2
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	2015.2
options.value	number   Date   string   array   boolean true   false	required	<p>The value to set the field to.</p> <p>The value type must correspond to the field type being set. For example:</p> <ul style="list-style-type: none"> <li>■ Text, Radio and Select fields accept string values.</li> <li>■ Checkbox fields accept Boolean values.</li> <li>■ Date and DateTime fields accept Date values.</li> <li>■ Integer, Float, Currency and Percent fields accept number values.</li> </ul>	2015.2

## Errors

Error Code	Thrown If
INVALID_FLD_VALUE	The options.value type does not match the field type.
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 script example, see [N/record Module Script Samples](#).

```
// Add additional code.
...
objRecord.setMatrixSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 12,
    line: 3,
    value: true
});
...
// Add additional code.
```

## Record.setSublistText(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value of a sublist field by a text representation. (standard mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )  When working in dynamic mode, set a sublist field text using the following methods: <ol style="list-style-type: none"><li>1. <a href="#">Record.selectLine(options)</a></li><li>2. <a href="#">Record.setCurrentSublistText(options)</a></li><li>3. <a href="#">Record.commitLine(options)</a></li></ol> Sets a string value with a "%" for rate and ratehighprecision fields.
<b>Returns</b>	<a href="#">record.Record</a>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	2015.2
options.text	string	required	The text to set the value to.	2015.2

### Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

Error Code	Thrown If
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax

 **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.
...
objRecord.setSublistText({
    sublistId: 'item',
    fieldId: 'item',
    line: 3,
    text: 'value'
});
...
// Add additional code.
```

## Record.setSublistValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value of a sublist field. (standard mode only — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )  When working in dynamic mode, set a sublist field value using the following methods: <ol style="list-style-type: none"> <li>1. Record.selectLine(options)</li> <li>2. Record.setCurrentSublistValue(options)</li> <li>3. Record.commitLine(options)</li> </ol> <p> <b>Important:</b> 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.</p>
<b>Returns</b>	record.Record
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	The internal ID of the sublist.  This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .	2015.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
options.line	number	required	The line number of the sublist. Note that line indexing begins at 0 with SuiteScript 2.0.	2015.2
options.value	number   Date   string   array   boolean <b>true</b>   <b>false</b>	required	The value to set the sublist field to.  The value type must correspond to the field type being set. For example: <ul style="list-style-type: none"> <li>■ Text, Radio and Select fields accept string values.</li> <li>■ Checkbox fields accept Boolean values.</li> <li>■ Date and DateTime fields accept Date values.</li> <li>■ Integer, Float, Currency and Percent fields accept number values.</li> </ul>	2015.2

## Errors

Error Code	Thrown If
INVALID_FLD_VALUE	The <code>options.value</code> type does not match the field type.
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_SUBLIST_OPERATION	A required argument is invalid or the sublist is not editable.

## Syntax

 <b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/record Module Script Samples</a> .
--

```
// Add additional code.
...
objRecord.setSublistValue({
  sublistId: 'item',
  fieldId: 'item',
  line: 3,
```

```

    value: true
});
...
// Add additional code.

```

## Record.setText(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value of the field by a text representation.  (dynamic and standard mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )  Sets a string value with a "%" for rate and ratehighprecision fields.
<b>Returns</b>	<a href="#">record.Record</a>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	The internal ID of a standard or custom body field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2
options.text	string   array	required	The text or texts to change the field value to. <ul style="list-style-type: none"> <li>■ If the field type is <b>multiselect</b>: <ul style="list-style-type: none"> <li>□ This parameter accepts an array of string values.</li> <li>□ This parameter accepts a null value. Passing in null deselects all currently selected values.</li> </ul> </li> <li>■ If the field type is <b>not multiselect</b>, this parameter accepts only a single string value.</li> </ul>	2015.2
options.ignoreFieldChange	boolean <code>true</code>   <code>false</code>	optional	If set to <code>true</code> , the field change and slaving event is ignored.  By default, this value is <code>false</code> .	2015.2

## Errors

Error Code	Thrown If
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 script example, see [N/record Module Script Samples](#).

```
// Add additional code.
...
objRecord.setText({
  fieldId: 'item',
  text: 'value',
  ignoreFieldChange: true
});
...
// Add additional code.
```

## Record.setValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value of a field.  (dynamic and standard mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Returns</b>	record.Record
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	The internal ID of a standard or custom body field.  See, <a href="#">How do I find a field's internal ID?</a>	2015.2

Parameter	Type	Required / Optional	Description	Since
<code>options.value</code>	number   Date   string   array   boolean <code>true</code>   <code>false</code>	required	<p>The value to set the field to.</p> <p>The value type must correspond to the field type being set. For example:</p> <ul style="list-style-type: none"> <li>■ Text, Radio, Select and Multi-Select fields accept string and array of values.</li> <li>■ Checkbox fields accept Boolean values.</li> <li>■ Date and DateTime fields accept Date values.</li> <li>■ Integer, Float, Currency and Percent fields accept number values.</li> </ul>	2015.2
<code>options.ignoreFieldChange</code>	boolean <code>true</code>   <code>false</code>	optional	<p>If set to <code>true</code>, the field change and slaving event is ignored.</p> <p>By default, this value is <code>false</code>.</p>	2015.2

## Errors

Error Code	Thrown If
INVALID_FLD_VALUE	The <code>options.value</code> type does not match the field type.
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 script example, see [N/record Module Script Samples](#).

```
// Add additional code.
...
objRecord.setValue({
  fieldId: 'item',
  value: true,
  ignoreFieldChange: true
});
...
// Add additional code.
```

## Record.id



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal ID of a specific record.
-----------------------------	---------------------------------------

	This property is not available to subrecords.  (dynamic and standard mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> )
<b>Type</b>	number (read-only)
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	2015.2

## Record.isDynamic



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>Indicates whether the record is in dynamic or standard mode.</p> <ul style="list-style-type: none"> <li>■ If set to <code>true</code>, the record is currently in dynamic mode. If set to <code>false</code>, the record is currently in standard mode.           <ul style="list-style-type: none"> <li>□ 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 <a href="#">Record.save(options)</a>.  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.</li> <li>□ 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.</li> </ul> </li> </ul> <p>This value is set when the record is created or accessed.</p> <p>(dynamic and standard mode — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a>)</p>
<b>Type</b>	boolean <code>true</code>   <code>false</code> (read-only)
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	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](#).

```
// Add additional code.  
...  
if (record.isDynamic) {  
    ...  
}  
...  
// Add additional code.
```

## Record.type



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The record type. Note the following: <ul style="list-style-type: none"> <li>■ When working with an instance of a standard NetSuite record type, set this value by using the <a href="#">record.Type</a> enum.</li> <li>■ 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 <a href="#">Custom Record</a>.</li> </ul> <p>This property is not available to subrecords. (dynamic and standard modes — see the help topic <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a>)</p>
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Record Object Members</a>
<b>Since</b>	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](#).

```
// 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

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a sublist on a standard or custom record.  For a complete list of this object's methods and properties, see <a href="#">Sublist Object Members</a> .
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	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](#).

```

// 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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a column in the sublist.
<b>Returns</b>	<a href="#">record.Column</a>
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Sublist Object Members</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	The internal ID of the column field in the sublist.  See, <a href="#">How do I find a field's internal ID?</a>	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](#).

### Example 1

```
// 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.

```
// 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the internal ID of the sublist.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Sublist Object Members</a>
<b>Since</b>	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](#).

```
// 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.

```

## Sublist.isChanged



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether the sublist has changed on the record form.
<b>Type</b>	boolean <code>true</code>   <code>false</code> (read-only)
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	<a href="#">Sublist Object Members</a>
<b>Since</b>	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](#).

```

// Add additional code.
...
var objRecord = record.load({
    type: record.Type.SALES_ORDER,
    id: 275
});

var objSublist = objRecord.getSublist({
    sublistId: 'item'
});

if(objSublist.isChanged){
    //Perform an action when objSublist.isChanged is true
}
...
// Add additional code.

```

## Sublist.isDisplay



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether the sublist is displayed on the record form.
-----------------------------	--

<b>Type</b>	boolean <code>true</code>   <code>false</code> (read-only)
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/record Module
<b>Sibling Object Members</b>	<a href="#">Sublist Object Members</a>
<b>Since</b>	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](#).

```
// 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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the sublist type. For more information on sublist types, see <a href="#">serverWidget.SublistType</a> .
	 <b>Important:</b> Sublist.type will return a lower case string representing the sublist type. For example, <code>inlineeditor</code> not <code>INLINEEDITOR</code> .
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/record Module</a>
<b>Sibling Object Members</b>	<a href="#">Sublist Object Members</a>
<b>Since</b>	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](#).

```
// 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)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Attaches a record to another record.
	<p><b>Note:</b> For the promise version of this method, see <a href="#">record.attach.promise(options)</a>. Note that promises are only supported in client scripts.</p>
<b>Returns</b>	void
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.record	<a href="#">record.Record</a>	required	The record to attach.	2015.2
options.record.type	string	required	The type of record to attach.	2015.2

Parameter	Type	Required / Optional	Description	Since
			Set this value using the <a href="#">record.Type</a> enum.  To attach a file from the file cabinet to a record, set type to <b>file</b> .	
<code>options.record.id</code>	number   string	required	The internal ID of the record to attach.	2015.2
<code>options.to</code>	<a href="#">record.Record</a>	required	The record that the <code>options.record</code> gets attached to.	2015.2
<code>options.to.type</code>	string	required	The record type of the record to attach to.  Set the value using the <a href="#">record.Type</a> enum.  To attach a file from the file cabinet to a record, set type to <b>file</b> .	2015.2
<code>options.to.id</code>	number   string	required	The internal ID of the record to attach to.	2015.2
<code>options.attributes</code>	Object	optional	The name-value pairs containing attributes for the attachment.  By default, this value is null.	2015.2

## Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	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](#).

```
// Add additional code.
...
var id = record.attach({
    record: {
        type: 'file',
        id: '200'
    },
    to: {
        type: 'customer',
        id: '90'
    }
});
...
// Add additional code.
```

```
record.attach({
```

```

    record: {
      type: 'contact',
      id: 2},
    to: {
      type: 'customer',
      id: 3},
    attributes: {
      role: 3}
  });

```

## record.attach.promise(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Attaches a record asynchronously to another record.   <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">record.attach(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Supported Script Types</b>	Client-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.record	<a href="#">record.Record</a>	required	The record to attach.	2015.2
options.record.type	string	required	The type of record to attach.  Set the value using the <a href="#">record.Type</a> enum.  To attach a file from the file cabinet to a record, set type to <b>file</b> .	2015.2
options.record.id	number   string	required	The internal ID of the record to attach.	2015.2
options.to	<a href="#">record.Record</a>	required	The record that the <b>options.record</b> gets attached to.	2015.2
options.to.type	string	required	The record type of the record to attach to.	2015.2

Parameter	Type	Required / Optional	Description	Since
			Set the value using the <code>record.Type</code> enum.  To attach a file from the file cabinet to a record, set type to <code>file</code> .	
<code>options.to.id</code>	number   string	required	The internal ID of the record to attach to.	2015.2
<code>options.attributes</code>	Object	optional	The name-value pairs containing attributes for the attachment.  By default, this value is null.	2015.2

## Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	A required argument is missing or undefined.

## Syntax

 **Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// 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'
        });

        }, function(e) {
        log.error({
            title: e.name,
            details: e.message
        });
    });
}
```

```

    });
});
}
...
// Add additional code.

```

## record.copy(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a new record by copying an existing record in NetSuite.   <b>Note:</b> For the promise version of this method, see <a href="#">record.copy.promise(options)</a> . Note that promises are only supported in client scripts.
<b>Returns</b>	<a href="#">record.Record</a>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	Transaction records: 10 usage units  Custom records: 2 usage units  All other records: 5 usage units
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.type</code>	string	required	<p>The record type.  Use the following guidelines:</p> <ul style="list-style-type: none"> <li>■ When copying an instance of a standard NetSuite record type, set this value by using the <a href="#">record.Type</a> enum.</li> <li>■ 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 <a href="#">Custom Record</a>.</li> </ul>	2015.2
<code>options.id</code>	number	required	The internal ID of the existing record instance in NetSuite.	2015.2
<code>options.isDynamic</code>	boolean <code>true</code>   <code>false</code>	optional	Determines whether the new record is created in dynamic mode.	2015.2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> <li>■ If set to <code>true</code>, the new record is created in dynamic mode.</li> <li>■ If set to <code>false</code>, the new record is created in standard mode.</li> </ul> <p>By default, this value is <code>false</code>.</p> <p><b>Note:</b> For additional information on standard and dynamic mode, see <a href="#">record.Record</a> and <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a>.</p>	
<code>options.defaultValues</code>	Object	optional	<p>Name-value pairs containing default values of fields in the new record.</p> <p>By default, this value is null.</p> <p>For a list of available record default values, see <a href="#">N/record Default Values</a> in the NetSuite Help Center.</p>	2015.2

## Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	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](#).

```
// 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)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a new record asynchronously by copying an existing record in NetSuite.
---------------------------	--

	<b>Note:</b> The parameters and errors thrown for this method are the same as those for <code>record.copy(options)</code> . For more information on promises, see <a href="#">Promise Object</a> .
Returns	<a href="#">Promise Object</a>
Supported Script Types	Client-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
Governance	Transaction records: 10 usage units  Custom records: 2 usage units  All other records: 5 usage units
Module	<a href="#">N/record Module</a>
Since	2015.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
<code>options.type</code>	string	required	<p>The record type.  Use the following guidelines:</p> <ul style="list-style-type: none"> <li>■ When copying an instance of a standard NetSuite record type, set this value by using the <code>record.Type</code> enum.</li> <li>■ 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 <a href="#">Custom Record</a>.</li> </ul>	2015.2
<code>options.id</code>	number	required	The internal ID of the existing record instance in NetSuite.	2015.2
<code>options.isDynamic</code>	boolean <code>true</code>   <code>false</code>	optional	<p>Determines whether the new record is created in dynamic mode.</p> <ul style="list-style-type: none"> <li>■ If set to <code>true</code>, the new record is created in dynamic mode.</li> <li>■ If set to <code>false</code>, the new record is created in standard mode.</li> </ul> <p>By default, this value is <code>false</code>.</p> <p><b>Note:</b> For additional information on standard and dynamic mode, see <a href="#">record.Record</a> and <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a>.</p>	2015.2
<code>options.defaultValues</code>	Object	optional	Name-value pairs containing default values of fields in the new record.	2015.2

Parameter	Type	Required / Optional	Description	Since
			By default, this value is null.	

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

## Syntax

**⚠ Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// 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
        });
    });
}
```

```

    }, function(e) {
        log.error({
            title: e.name,
            details: e.message
        });
    });
}
...
// Add additional code.

```

## record.create(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a new record.
	 <b>Note:</b> For the promise version of this method, see <a href="#">record.create.promise(options)</a> . Note that promises are only supported in client scripts.
<b>Returns</b>	<a href="#">record.Record</a>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	Transaction records: 10 usage units  Custom records: 2 usage units  All other records: 5 usage units
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	<p>The record type.</p> <p>This value determines the <a href="#">Record.type</a> property of the record that is created. This property is read-only on an existing record.</p> <p>Use the following guidelines:</p> <ul style="list-style-type: none"> <li>■ When creating an instance of a standard NetSuite record type, set this value by using the <a href="#">record.Type</a> enum.</li> <li>■ When creating an instance of a custom record type, set this value by using the</li> </ul>	2015.2

Parameter	Type	Required / Optional	Description	Since
			custom record type's string ID. For help finding this ID, see the help topic <a href="#">Custom Record</a> .	
options.isDynamic	boolean <code>true</code>   <code>false</code>	optional	<p>Determines whether the new record is created in dynamic mode.</p> <ul style="list-style-type: none"> <li>■ If set to <code>true</code>, the new record is created in dynamic mode.</li> <li>■ If set to <code>false</code>, the new record is created in standard mode.</li> </ul> <p>By default, this value is <code>false</code>.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <span style="color: #0070C0; font-size: 1.2em; border-radius: 50%; width: 1.2em; height: 1.2em; display: inline-block; vertical-align: middle;"></span> <b>Note:</b> For additional information on standard and dynamic mode, see <a href="#">record.Record</a> and <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a>.       </div>	2015.2
options.defaultValues	Object	optional	<p>Name-value pairs containing default values of fields in the new record.</p> <p>By default, this value is null.</p> <p>For a list of available record default values, see <a href="#">N/record Default Values</a> in the NetSuite Help Center.</p>	2015.2

## Errors

Error Code	Thrown If
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 script example, see [N/record Module Script Samples](#).

```
// 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
});

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

## record.create.promise(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a new record asynchronously.  <span style="border: 1px solid #0070C0; border-radius: 15px; padding: 2px 10px; color: #0070C0;">(i) Note:</span> The parameters and errors thrown for this method are the same as those for <a href="#">record.create(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	<a href="#">Promise Object</a>
<b>Supported Script Types</b>	Client-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	Transaction records: 10 usage units  Custom records: 2 usage units  All other records: 5 usage units
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	<p>The record type.</p> <p>This value determines the <a href="#">Record.type</a> property of the record that is created. This property is read-only on an existing record.</p> <p>Use the following guidelines:</p> <ul style="list-style-type: none"> <li>■ When creating an instance of a standard NetSuite record type, set this value by using the <a href="#">record.Type</a> enum.</li> <li>■ 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 <a href="#">Custom Record</a>.</li> </ul>	2015.2

Parameter	Type	Required / Optional	Description	Since
<code>options.isDynamic</code>	boolean <code>true</code>   <code>false</code>	optional	<p>Determines whether the new record is created in dynamic mode.</p> <ul style="list-style-type: none"> <li>▪ If set to <code>true</code>, the new record is created in dynamic mode.</li> <li>▪ If set to <code>false</code>, the new record is created in standard mode.</li> </ul> <p>By default, this value is <code>false</code>.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <span style="color: #0070C0; font-size: 1.5em; border-radius: 50%; width: 1.2em; height: 1.2em; display: inline-block; vertical-align: middle;"></span> <b>Note:</b> For additional information on standard and dynamic mode, see <a href="#">record.Record</a> and <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a>.       </div>	2015.2
<code>options.defaultValues</code>	Object	optional	<p>Name-value pairs containing default values of fields in the new record.</p> <p>By default, this value is null.</p>	2015.2

## Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	A required argument is missing or undefined.

## Syntax



**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// Add additional code.

...
function createRecord() {

  // Create an instance of a standard record 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({
        title: 'New record saved',
        details: 'New record ID: ' + recordId
    });

}, function(e) {
    log.error({
        title: 'Unable to create record',
        details: e.name
    });
});
}
...
// Add additional code.

```

## record.delete(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Deletes a record.  <b>i Note:</b> For the promise version of this method, see <a href="#">record.delete.promise(options)</a> . Note that promises are only supported in client scripts.
<b>Returns</b>	The internal ID of the deleted <a href="#">record.Record</a> .
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	Transaction records: 20 usage units  Custom records: 4 usage units  All other records: 10 usage units
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The record type.	2015.2

Parameter	Type	Required / Optional	Description	Since
			<p>Use the following guidelines:</p> <ul style="list-style-type: none"> <li>■ When deleting an instance of a standard NetSuite record type, set this value by using the <a href="#">record.Type</a> enum.</li> <li>■ 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>.</li> </ul>	
<code>options.id</code>	number   string	required	The internal ID of the record instance to be deleted.	2015.2

## Errors

Error Code	Thrown If
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 script example, see [N/record Module Script Samples](#).

```
// 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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Deletes a record asynchronously.
---------------------------	----------------------------------

	<b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">record.delete(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Supported Script Types</b>	Client-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	Transaction records: 20 usage units Custom records: 4 usage units All other records: 10 usage units
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	2015.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The record type.  Use the following guidelines: <ul style="list-style-type: none"><li>■ When deleting an instance of a standard NetSuite record type, set this value by using the <a href="#">record.Type enum</a>.</li><li>■ 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>.</li></ul>	2015.2
options.id	number   string	required	The internal ID of the record instance to be deleted.	2015.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

## Syntax

<b>Important:</b> The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see <a href="#">Promise Object</a> .
---

```
// 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() {

        log.debug({
            title: 'Success',
            details: 'Record successfully deleted'
        });

        // Add any other needed code that should execute
        // after the record is deleted.

    }, function(e) {
        log.error({
            title: 'Unable to delete record',
            details: e.name
        });
    });
}

...
// Add additional code

```

## record.detach(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Detaches a record from another record.
	 <b>Note:</b> For the promise version of this method, see <a href="#">record.detach.promise(options)</a> . Note that promises are only supported in client scripts.
<b>Returns</b>	void
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
<code>options.record</code>	<code>record.Record</code>	required	The record to be detached.	2015.2
<code>options.record.type</code>	string	required	The type of record to be detached. Set this value using the <code>record.Type</code> enum.	2015.2
<code>options.record.id</code>	number   string	required	The ID of the record to be detached.	2015.2
<code>options.from</code>	<code>record.Record</code>	required	The destination record that <code>options.record</code> should be detached from.	2015.2
<code>options.from.type</code>	string	required	The type of the destination. Set this value using the <code>record.Type</code> enum.	2015.2
<code>options.from.id</code>	number   string	required	The ID of the destination.	2015.2
<code>options.attributes</code>	Object	optional	Name-value pairs containing default values of fields in the new record.  By default, this value is null.	2015.2

## Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	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](#).

```
// Add additional code.
...
record.detach({
    record: {
        type: 'file',
        id:'200'
    },
    from: {
        type: 'customer',
        id:'90'
    }
})
...
// Add additional code.
```

## record.detach.promise(options)

<b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--------------	---

<b>Method Description</b>	Detaches a record from another record asynchronously.  <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">record.detach(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	<a href="#">Promise Object</a>
<b>Supported Script Types</b>	Client-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	2015.2

### Parameters

<b>Note:</b>	The options parameter is a JavaScript object.
--------------	---

Parameter	Type	Required / Optional	Description	Since
options.record	<a href="#">record.Record</a>	required	The record to be detached.	2015.2
options.record.type	string	required	The type of record to be detached.  Set this value using the <a href="#">record.Type</a> enum.	2015.2
options.record.id	number   string	required	The ID of the record to be detached.	2015.2
options.from	<a href="#">record.Record</a>	required	The destination record that <b>options.record</b> should be detached from.	2015.2
options.from.type	string	required	The type of the destination.  Set this value using the <a href="#">record.Type</a> enum.	2015.2
options.from.id	number   string	required	The ID of the destination.	2015.2
options.attributes	Object	optional	Name-value pairs containing default values of fields in the new record.  By default, this value is null.	2015.2

### Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

## Syntax



**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// 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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Loads an existing record.
	<p><b>Note:</b> For the promise version of this method, see <a href="#">record.load.promise(options)</a>. Note that promises are only supported in client scripts. Make sure to save the record before loading it.</p>
<b>Returns</b>	record.Record

<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	Transaction records: 10 usage units Custom records: 2 usage units All other records: 5 usage units
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	<p>The record type. Use the following guidelines:</p> <ul style="list-style-type: none"> <li>■ When loading an instance of a standard NetSuite record type, set this value by using the <code>record.Type</code> enum.</li> <li>■ 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 <a href="#">Custom Record</a>.</li> </ul>	2015.2
options.id	number	required	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.	
options.isDynamic	boolean <code>true</code>   <code>false</code>	optional	<p>Determines whether the record is loaded in dynamic mode.</p> <ul style="list-style-type: none"> <li>■ If set to <code>true</code>, the record is loaded in dynamic mode.</li> <li>■ If set to <code>false</code>, the record is loaded in standard mode.</li> </ul> <p>By default, this value is <code>false</code>.</p> <p> <b>Note:</b> For additional information on standard and dynamic mode, see <a href="#">record.Record</a> and <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a>.</p>	2015.2
options.defaultValues	Object	optional	<p>Name-value pairs containing default values of fields in the new record. By default, this value is null. For a list of available record default values, see <a href="#">N/record Default Values</a> in the NetSuite Help Center.</p>	

## Errors

Error Code	Thrown If
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 script example, see [N/record Module Script Samples](#).

```
// 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)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Loads an existing record asynchronously.
	<p><b>i Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">record.load(options)</a>. For more information on promises, see <a href="#">Promise Object</a>.</p>
<b>Returns</b>	Promise Object
<b>Supported Script Types</b>	Client-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	Transaction records: 10 usage units Custom records: 2 usage units All other records: 5 usage units
<b>Module</b>	<a href="#">N/record Module</a>

<b>Since</b>	2015.2
--------------	--------

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The record type.  Use the following guidelines: <ul style="list-style-type: none"><li>■ When loading an instance of a standard NetSuite record type, set this value by using the <code>record.Type</code> enum.</li><li>■ 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 <a href="#">Custom Record</a>.</li></ul>	2015.2
options.id	number	required	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.	
options.isDynamic	boolean <code>true</code>   <code>false</code>	optional	Determines whether the record is loaded in dynamic mode. <ul style="list-style-type: none"><li>■ If set to <code>true</code>, the record is loaded in dynamic mode.</li><li>■ If set to <code>false</code>, the record is loaded in standard mode.</li></ul> By default, this value is <code>false</code> .  <b>Note:</b> For additional information on standard and dynamic mode, see <a href="#">record.Record</a> and <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a> .	2015.2
options.defaultValues	Object	optional	Name-value pairs containing default values of fields in the new record.  By default, this value is null.	

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

## Syntax

<b>Important:</b> The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see <a href="#">Promise Object</a> .
---

```
function loadRecord() {
```

```

// 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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Updates and submits one or more body fields on an existing record in NetSuite, and returns the internal ID of the parent record.</p> <p>When you use this method, you do not need to load or submit the parent record.</p> <p>You <b>can</b> use this method to edit and submit the following:</p> <ul style="list-style-type: none"> <li>■ Standard body fields that support inline editing (direct list editing). For more information, see the help topic <a href="#">Using Inline Editing</a>.</li> <li>■ Custom body fields that support inline editing.</li> </ul> <p>You <b>cannot</b> use this method to edit and submit the following:</p>
---------------------------	--

	<ul style="list-style-type: none"> <li>■ Select fields</li> <li>■ Sublist line item fields</li> <li>■ Subrecord fields (for example, address fields)</li> </ul> <p><b>Note:</b> For the promise version of this method, see <a href="#">record.submitFields.promise(options)</a>. Note that promises are only supported in client scripts.</p>
<b>Returns</b>	The internal ID of the parent record.
<b>Supported Script Types</b>	Client and server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	Transaction records: 10 usage units Custom records: 2 usage units All other records: 5 usage units
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	2015.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	<p>The record type. Use the following guidelines:</p> <ul style="list-style-type: none"> <li>■ When working with an instance of a standard NetSuite record type, set this value by using the <a href="#">record.Type</a> enum.</li> <li>■ 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 <a href="#">Custom Record</a>.</li> </ul>	2015.2
options.id	number   string	required	The internal ID of the existing record instance in NetSuite.	2015.2
options.values	Object	required	<p>The ID-value pairs for each field you want to edit and submit. The value type must correspond to the field type being set. For example:</p> <ul style="list-style-type: none"> <li>■ Text, Radio, Select and Multi-Select fields accept string values.</li> <li>■ Checkbox fields accept Boolean values.</li> <li>■ Date and DateTime fields accept Date values.</li> <li>■ Integer, Float, Currency and Percent fields accept number values.</li> </ul>	2015.2

Parameter	Type	Required / Optional	Description	Since
options.options	Object	optional	Additional options to set for the record.	2015.2
options.options.enableSourcing	boolean <code>true</code>   <code>false</code>	optional	Indicates whether to enable sourcing during the record update.  By default, this value is <code>true</code> .	2015.2
options.options.ignoreMandatoryFields	boolean <code>true</code>   <code>false</code>	optional	Indicates whether to ignore mandatory fields during record submission.  By default, this value is <code>false</code> .	2015.2

## Errors

Error Code	Thrown If
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 script example, see [N/record Module Script Samples](#).

```
// 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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Updates and submits one or more body fields asynchronously on an existing record in NetSuite, and returns the internal ID of the parent record.</p> <p>When you use this method, you do not need to load or submit the parent record.</p> <p>You <b>can</b> use this method to edit and submit the following:</p> <ul style="list-style-type: none"> <li>■ Standard body fields that support inline editing (direct list editing). For more information, see the help topic <a href="#">Using Inline Editing</a>.</li> <li>■ Custom body fields that support inline editing.</li> </ul> <p>You <b>cannot</b> use this method to edit and submit the following:</p> <ul style="list-style-type: none"> <li>■ Select fields</li> <li>■ Sublist line item fields</li> <li>■ Subrecord fields (for example, address fields)</li> </ul> <p> <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">record.submitFields(options)</a>. For more information on promises, see <a href="#">Promise Object</a>.</p>
<b>Returns</b>	Promise Object
<b>Supported Script Types</b>	<p>Client-side scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a>.</p>
<b>Governance</b>	<p>Transaction records: 10 usage units</p> <p>Custom records: 2 usage units</p> <p>All other records: 5 usage units</p>
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	<p>The record type.</p> <p>Use the following guidelines:</p> <ul style="list-style-type: none"> <li>■ When working with an instance of a standard NetSuite record type, set this value by using the <a href="#">record.Type</a> enum.</li> <li>■ 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 <a href="#">Custom Record</a>.</li> </ul>	2015.2

Parameter	Type	Required / Optional	Description	Since
options.id	number   string	required	The internal ID of the existing record instance in NetSuite.	2015.2
options.values	Object	required	The ID-value pairs for each field you want to edit and submit.	2015.2
options.options	Object	optional	Additional options to set for the record.	2015.2
options.options.enableSourcing	boolean <b>true</b>   <b>false</b>	optional	Indicates whether to enable sourcing during the record update. By default, this value is <b>true</b> .	2015.2
options.options.ignoreMandatoryFields	boolean <b>true</b>   <b>false</b>	optional	Indicates whether to ignore mandatory fields during record submission. By default, this value is <b>false</b> .	2015.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

## Syntax

**⚠ Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// 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,
        });
    });
}
```

```

        details: e.message
    });
}
...
// Add additional code

```

## record.transform(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">Supported Transformation Types</a> .
<b>Returns</b>	<a href="#">record.Record</a>
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	Transaction records: 10 usage units  Custom records: 2 usage units  All other record types: 5 usage units
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.fromType	string	required	The record type of the existing record instance being transformed.  This value sets the <a href="#">Record.type</a> property for the record. This property is read-only and cannot be changed after the record is loaded.  Set this value using the <a href="#">record.Type</a> .	2015.2
options.fromId	number	required	The internal ID of the existing record instance being transformed.	2015.2
options.toType	string	required	The record type of the record returned when the transformation is complete.	2015.2

Parameter	Type	Required / Optional	Description	Since
<code>options.isDynamic</code>	boolean <code>true</code>   <code>false</code>	optional	<p>Determines whether the new record is created in dynamic mode.</p> <ul style="list-style-type: none"> <li>■ If set to <code>true</code>, the new record is created in dynamic mode.</li> <li>■ If set to <code>false</code>, the new record is created in standard mode.</li> </ul> <p>By default, this value is <code>false</code>.</p> <p><b>Note:</b> For additional information on standard and dynamic mode, see <a href="#">record.Record</a> and <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a>.</p>	2015.2
<code>options.defaultValues</code>	Object	optional	<p>Name-value pairs containing default values of fields in the new record.</p> <p>By default, this value is null.</p> <p>For a list of available record default values, see <a href="#">N/record Default Values</a> in the NetSuite Help Center.</p>	2015.2

## Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	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](#).

```
// Add additional code.
...
var objRecord = record.transform({
    fromType: record.Type.CUSTOMER,
    fromId: 107,
    toType: record.Type.SALES_ORDER,
    isDynamic: true,
});
...
// Add additional code.
```

```
// Add additional code.
...
record.transform({
    fromType:'salesorder',
    fromId: 6,
    toType: 'invoice',
    defaultValues: {
```

```

    billdate: '01/01/2019'} });
...
// Add additional code.

```

## Supported Transformation Types

Original Record Type	Transformed Record Type
Build/Assembly	Assembly Build
Assembly Build	Assembly Unbuild
Cash Sale	Cash Sale
Customer	Cash Sale
Customer	Customer Payment
	<p> <b>Note:</b> When you use this transformation, do not use <a href="#">Record.setValue(options)</a> to set the value of the <code>customer</code> field on the resulting Customer Payment record. This field is populated automatically during transformation, and you cannot specify a value for this field after the transformation is complete.</p>
Customer	Quote
Customer	Invoice
Customer	Opportunity
Customer	Sales Order
Employee	Expense Report
Employee	Time
Quote	Cash Sale
Quote	Invoice
Quote	Sales Order
Invoice	Credit Memo
Invoice	Customer Payment
Invoice	Return Authorization
Lead	Opportunity
Opportunity	Cash Sale
Opportunity	Quote
Opportunity	Invoice
Opportunity	Sales Order
Prospect	Quote
Prospect	Opportunity
Prospect	Sales Order

Original Record Type	Transformed Record Type
Purchase Order	Item Receipt
Purchase Order	Vendor Bill
Purchase Order	Vendor Return Authorization
Return Authorization	Cash Refund
Return Authorization	Credit Memo
Return Authorization	Item Receipt
Return Authorization	Revenue Commitment Reversal
<b>Note:</b> The return authorization must be approved and received for this transformation to work.	
Sales Order	Cash Sale
Sales Order	Invoice
Sales Order	Item Fulfillment
Sales Order	Return Authorization
Sales Order	Revenue Commitment
Transfer Order	Item Fulfillment
Transfer Order	Item Receipt
Vendor	Purchase Order
Vendor	Vendor Bill
Vendor Bill	Vendor Credit
Vendor Bill	Vendor Payment
Vendor Bill	Vendor Return Authorization
Vendor Return Authorization	Item Fulfillment
Vendor Return Authorization	Vendor Credit
Work Order	Assembly Build

## record.transform.promise(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Transforms a record from one type into another asynchronously, using data from an existing record.</p> <p>You can use this method to automate order processing, creating item fulfillment transactions and invoices off of orders.</p>
---------------------------	---

	For a list of supported transformations, see <a href="#">Supported Transformation Types</a> .
	<p><b>Note:</b> The parameters and errors thrown for this method are the same as those for <code>record.transform(options)</code>. For more information on promises, see <a href="#">Promise Object</a>.</p>
<b>Returns</b>	Promise Object
<b>Supported Script Types</b>	Client-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	Transaction records: 10 usage units Custom records: 2 usage units All other record types: 5 usage units
<b>Module</b>	<a href="#">N/record Module</a>
<b>Since</b>	2015.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
<code>options.fromType</code>	string	required	The record type of the existing record instance being transformed.  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.  Set this value using the <code>record.Type</code> .	2015.2
<code>options.fromId</code>	number	required	The internal ID of the existing record instance being transformed.	2015.2
<code>options.toType</code>	string	required	The record type of the record returned when the transformation is complete.	2015.2
<code>options.isDynamic</code>	boolean <code>true</code>   <code>false</code>	optional	<p>Determines whether the new record is created in dynamic mode.</p> <ul style="list-style-type: none"> <li>▪ If set to <code>true</code>, the new record is created in dynamic mode.</li> <li>▪ If set to <code>false</code>, the new record is created in standard mode.</li> </ul> <p>By default, this value is <code>false</code>.</p> <p><b>Note:</b> For additional information on standard and dynamic mode, see <code>record.Record</code> and <a href="#">SuiteScript 2.0 – Standard and Dynamic Modes</a>.</p>	2015.2
<code>options.defaultValues</code>	Object	optional	Name-value pairs containing default values of fields in the new record.	2015.2

Parameter	Type	Required / Optional	Description	Since
			By default, this value is null.	

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.

## Syntax

**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// Add additional code.
...
function transformRecord() {

    var transformRecordPromise = record.transform.promise({
        fromType: record.Type.estimate,
        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

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for supported record types.  This enum is used to set the value of the <a href="#">Record.type</a> 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 <a href="#">Record.type</a> property by using the custom record type's string ID. For more help finding this ID, see the help topic <a href="#">Custom Record</a> .)
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/record Module
<b>Since</b>	2015.2

### Values

■ ACCOUNT	■ EMAIL_TEMPLATE	■ PRICE_BOOK
■ ACCOUNTING_BOOK	■ EMPLOYEE	■ PRICE_LEVEL
■ ACCOUNTING_CONTEXT	■ EMPLOYEE_CHANGE_REQUEST	■ PRICE_PLAN
■ ACCOUNTING_PERIOD	■ EMPLOYEE_CHANGE_TYPE	■ PRICING_GROUP
■ ADV_INTER_COMPANY_JOURNAL_ENTRY	■ ENTITY_ACCOUNT_MAPPING	■ PROJECT_EXPENSE_TYPE
■ ALLOCATION_SCHEDULE	■ ESTIMATE	■ PROJECT_TASK
■ AMORTIZATION_SCHEDULE	■ EXPENSE_AMORTIZATION_EVENT	■ PROJECT_TEMPLATE
■ AMORTIZATION_TEMPLATE	■ EXPENSE_CATEGORY	■ PROMOTION_CODE
■ AS_CHARGED_PROJECT_REVENUE_RULE	■ EXPENSE_PLAN	■ PROSPECT
■ ASSEMBLY_BUILD	■ EXPENSE_REPORT	■ PURCHASE_CONTRACT
■ ASSEMBLY_ITEM	■ FAIR_VALUE_PRICE	■ PURCHASE_ORDER
■ ASSEMBLY_UNBUILD	■ FIXED_AMOUNT_PROJECT_REVENUE_RULE	■ PURCHASE_REQUSITION
■ BILLING_ACCOUNT	■ FOLDER	■ REALLOCATE_ITEM
■ BILLING_CLASS	■ FULFILLMENT_REQUEST	■ RECEIVE_INBOUND_SHIPMENT
■ BILLING_RATE_CARD	■ GENERAL_TOKEN	■ RESOURCE_ALLOCATION
■ BILLING_REVENUE_EVENT	■ GENERIC_RESOURCE	■ RESTLET
■ BILLING_SCHEDULE	■ GIFT_CERTIFICATE	■ RETURN_AUTHORIZATION
■ BIN	■ GIFT_CERTIFICATE_ITEM	■ REVENUE_ARRANGEMENT
■ BIN_TRANSFER	■ GLOBAL_ACCOUNT_MAPPING	■ REVENUE_COMMITMENT
■ BIN_WORKSHEET	■ GLOBAL_INVENTORY_RELATIONSHIP	■ REVENUE_COMMITMENT_REVERSAL
■ BLANKET_PURCHASE_ORDER	■ GL_NUMBERING_SEQUENCE	■ REVENUE_PLAN
■ BOM	■ GOAL	■ REV_REC_SCHEDULE
■ BOM_REVISION	■ INBOUND_SHIPMENT	■ REV_REC_TEMPLATE
■ BULK_OWNERSHIP_TRANSFER	■ INTERCOMP_ALLOCATION_SCHEDULE	■ SALES_ORDER
	■ INTER_COMPANY_JOURNAL_ENTRY	■ SALES_ROLE

■ BUNDLE_INSTALLATION_SCRIPT	■ INTER_COMPANY_TRANSFER_ORDER	■ SALES_TAX_ITEM
■ CALENDAR_EVENT	■ INVENTORY_ADJUSTMENT	■ SCHEDULED_SCRIPT
■ CAMPAIGN	■ INVENTORY_COST_REVALUATION	■ SCHEDULED_SCRIPT_INSTANCE
■ CAMPAIGN_RESPONSE	■ INVENTORY_COUNT	■ SCRIPT_DEPLOYMENT
■ CAMPAIGN_TEMPLATE	■ INVENTORY_DETAIL	■ SERIALIZED_ASSEMBLY_ITEM
■ CASH_REFUND	■ INVENTORY_ITEM	■ SERIALIZED_INVENTORY_ITEM
■ CASH_SALE	■ INVENTORY_NUMBER	■ SERVICE_ITEM
■ CHARGE	■ INVENTORY_STATUS	■ SHIP_ITEM
■ CHARGE_RULE	■ INVENTORY_STATUS_CHANGE	■ SOLUTION
■ CHECK	■ INVENTORY_TRANSFER	■ STATISTICAL_JOURNAL_ENTRY
■ CLASSIFICATION	■ INVOICE	■ STORE_PICKUP_FULFILLMENT
■ CLIENT_SCRIPT	■ ISSUE	■ SUBSCRIPTION
■ CMS_CONTENT	■ ISSUE_PRODUCT	■ SUBSCRIPTION_CHANGE_ORDER
■ CMS_CONTENT_TYPE	■ ISSUE_PRODUCT_VERSION	■ SUBSCRIPTION_LINE
■ CMS_PAGE	■ ITEM_ACCOUNT_MAPPING	■ SUBSCRIPTION_PLAN
■ COMMERCE_CATEGORY	■ ITEM_DEMAND_PLAN	■ SUBSIDIARY
■ COMPETITOR	■ ITEM_FULFILLMENT	■ SUBTOTAL_ITEM
■ CONSOLIDATED_EXCHANGE_RATE	■ ITEM_GROUP	■ SUITELET
■ CONTACT	■ ITEM_LOCATION_CONFIGURATION	■ SUPPLY_CHAIN_SNAPSHOT
■ CONTACT_CATEGORY	■ ITEM_RECEIPT	■ SUPPORT_CASE
■ CONTACT_ROLE	■ ITEM_REVISION	■ TASK
■ COST_CATEGORY	■ ITEM_SUPPLY_PLAN	■ TAX_ACCT
■ COUPON_CODE	■ JOB	■ TAX_GROUP
■ CREDIT_CARD_CHARGE	■ JOB_STATUS	■ TAX_PERIOD
■ CREDIT_CARD_REFUND	■ JOB_TYPE	■ TAX_TYPE
■ CREDIT_MEMO	■ JOURNAL_ENTRY	■ TERM
■ CURRENCY	■ KIT_ITEM	■ TIME_BILL
■ CUSTOMER	■ LABOR_BASED_PROJECT_REVENUE_RULE	■ TIME_ENTRY
■ CUSTOMER_CATEGORY	■ LEAD	■ TIME_OFF_CHANGE
■ CUSTOMER_DEPOSIT	■ LOCATION	■ TIME_OFF_PLAN
■ CUSTOMER_MESSAGE	■ LOT_NUMBERED_ASSEMBLY_ITEM	■ TIME_OFF_REQUEST
■ CUSTOMER_PAYMENT	■ LOT_NUMBERED_INVENTORY_ITEM	■ TIME_OFF_RULE
■ CUSTOMER_PAYMENT_AUTHORIZATION	■ MANUFACTURING_COST_TEMPLATE	■ TIME_OFF_TYPE
■ CUSTOMER_REFUND	■ MANUFACTURING_OPERATION_TASK	■ TIME_SHEET
■ CUSTOMER_STATUS	■ MANUFACTURING_ROUTING	■ TOPIC
■ CUSTOMER_SUBSIDIARY_RELATIONSHIP	■ MAP_REDUCE_SCRIPT	■ TRANSACTION
■ CUSTOM_RECORD	■ MARKUP_ITEM	■ TRANSFER_ORDER
■ CUSTOM_TRANSACTION	■ MASSUPDATE_SCRIPT	■ UNITS_TYPE
■ DEPARTMENT	■ MERCHANTISE_HIERARCHY_LEVEL	■ USAGE
■ DEPOSIT	■ MERCHANTISE_HIERARCHY_NODE	■ USEREVENT_SCRIPT
■ DEPOSIT_APPLICATION	■ MERCHANTISE_HIERARCHY_VERSION	■ VENDOR
■ DESCRIPTION_ITEM	■ MESSAGE	■ VENDOR_BILL
■ DISCOUNT_ITEM	■ MFG_PLANNED_TIME	■ VENDOR_CATEGORY
■ DOWNLOAD_ITEM	■ NEXUS	■ VENDOR_CREDIT
	■ NON_INVENTORY_ITEM	■ VENDOR_PAYMENT
	■ NOTE	■ VENDOR_RETURN_AUTHORIZATION
	■ NOTE_TYPE	

	<ul style="list-style-type: none"> <li>■ OPPORTUNITY</li> <li>■ ORDER_SCHEDULE</li> <li>■ OTHER_CHARGE_ITEM</li> <li>■ OTHER_NAME</li> <li>■ OTHER_NAME_CATEGORY</li> <li>■ PARTNER</li> <li>■ PARTNER_CATEGORY</li> <li>■ PARTNER_COMMISSION_PLAN</li> <li>■ PAYCHECK</li> <li>■ PAYCHECK_JOURNAL</li> <li>■ PAYMENT_CARD</li> <li>■ PAYMENT_CARD_TOKEN</li> <li>■ PAYMENT_ITEM</li> <li>■ PAYMENT_METHOD</li> <li>■ PAYROLL_ITEM</li> <li>■ PCT_COMPLETE_PROJECT_REVENUE_RULE</li> <li>■ PERFORMANCE REVIEW</li> <li>■ PERFORMANCE REVIEW_SCHEDULE</li> <li>■ PERIOD_END_JOURNAL</li> <li>■ PHONE_CALL</li> <li>■ PORTLET</li> </ul>	<ul style="list-style-type: none"> <li>■ VENDOR_SUBSIDIARY_RELATIONSHIP</li> <li>■ WEBSITE</li> <li>■ WORKFLOW_ACTION_SCRIPT</li> <li>■ WORK_ORDER</li> <li>■ WORK_ORDER_CLOSE</li> <li>■ WORK_ORDER_COMPLETION</li> <li>■ WORK_ORDER_ISSUE</li> <li>■ WORKPLACE</li> </ul>
--	--	---

## Syntax



**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 objRecord = record.delete({
    type: record.Type.SALES_ORDER,
    id: 128
});
...
// Add additional code.
```

## N/redirect Module



**Note:** The content in this help topic pertains to SuiteScript 2.0.

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 Samples

## N/redirect Module Members

Member Type	Name	Return Type	Supported Script Types	Description
Method	redirect.redirect(options)	void	Suitelets, beforeLoad user events, and synchronous afterSubmit user events	Redirects to the URL of a Suitelet that is available externally (available without login).
	redirect.toRecord(options)	void	Suitelets, beforeLoad user events, and synchronous afterSubmit user events	Redirects to a NetSuite record.
	redirect.toSavedSearch(options)	void	afterSubmit user events	Redirects to a saved search.
	redirect.toSavedSearchResult(options)	void	afterSubmit user events	Redirects to a saved search result.
	redirect.toSearch(options)	void	afterSubmit user events	Redirects to search.
	redirect.toSearchResult(options)	void	afterSubmit user events	Redirects to search results.
	redirect.toSuitelet(options)	void	Suitelets, beforeLoad user events, and synchronous afterSubmit user events	Redirects to a Suitelet.
	redirect.toTaskLink(options)	void	Suitelets, beforeLoad user events, and synchronous afterSubmit user events	Redirects to a tasklink.

## N/redirect Module Script Samples

The following script samples demonstrate how to use the features of the N/redirect module.

### Sample 1: Redirect to a task record



**Note:** This sample script uses the **require** function so that you can copy it into the SuiteScript Debugger and test it. You must use the **define** function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample sets the redirect URL to a newly created task record. To set a redirect using a record ID, the record must have been previously submitted to NetSuite.

```
/**
```

```

* @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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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).
<b>Returns</b>	Void
<b>Supported Script Types</b>	Suitelets, beforeLoad user events, and synchronous afterSubmit user events For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/redirect Module</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description
options.url	string	required	The URL of a Suitelet that is available externally  <div style="border: 1px solid #ccc; padding: 5px;"> <span style="color: #0070C0;">i</span> <b>Note:</b> For an external URL, Available without Login must be enabled on the Script Deployment page for the Suitelet.         </div>
options.parameters	Object	optional	Contains additional URL parameters as key/value pairs.

Parameter	Type	Required / Optional	Description
			<p><b>Note:</b> Parameters cannot be arrays. Use JSON.stringify/JSON.parse instead to handle arrays.</p>

## Syntax

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

```
//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)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to set the redirect URL to a specific NetSuite record.
	<p><b>Note:</b> 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.</p>
<b>Returns</b>	Void
<b>Supported Script Types</b>	Suitelets, beforeLoad user events, and synchronous afterSubmit user events For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/redirect Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.id	string	required	The internal id of the target record.
options.type	string	required	The type of record.

Parameter	Type	Required / Optional	Description
<code>options.isEditMode</code>	boolean <code>true</code>   <code>false</code>	optional	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 <code>false</code> – returns the URL to a record in view mode.
<code>options.parameters</code>	Object	optional	Contains additional URL parameters as key/value pairs.

## Syntax

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

```
//Add additional code
...
redirect.toRecord({
    type : record.Type.TASK,
    id : taskRecordId,
    parameters: {'custparam_test':'helloWorld'}
});
...
//Add additional code
```

## redirect.toSavedSearch(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to load an existing saved search and redirect to the populated search definition page.
<b>Returns</b>	Void
<b>Supported Script Types</b>	afterSubmit user event scripts For more information, see the help topic <a href="#">SuiteScript 2.0 User Event Script Type</a> .
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/redirect Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.id</code>	number	required	Internal ID of the search.

Parameter	Type	Required / Optional	Description
			<p>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>.</p> <p>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.</p>

## Syntax

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

```
//Add additional code
...
redirect.toSavedSearch({id: 234});
...
//Add additional code
```

## redirect.toSavedSearchResult(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to redirect a user to a search results page for an existing saved search.
<b>Returns</b>	Void
<b>Supported Script Types</b>	afterSubmit user event scripts For more information, see the help topic <a href="#">SuiteScript 2.0 User Event Script Type</a> .
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/redirect Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.id</code>	number	required	<p>Internal ID of the search.</p> <p>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>.</p> <p>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.</p>

## Syntax



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

```
//Add additional code
...
redirect.toSavedSearchResult({id: 234});
...
//Add additional code
```

## redirect.toSearch(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	Void
<b>Supported Script Types</b>	afterSubmit user event scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 User Event Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/redirect Module</a>
<b>Since</b>	2015.2

## Parameters



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

Parameter	Type	Required / Optional	Description
options.search	<a href="#">search.Search</a>	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 redirect Module Script Sample.

```
var column = ['internalid'];
var filter = [["mainline", "is", "T"]];
var ourNewSearch = search.create({
    id: 'customsearch_test',
    type: search.Type.SALES_ORDER,
    title: 'My Generated Search',
    columns: column,
    filters: filter
});
```

```
redirect.toSearch({
  search: ourNewSearch
});
```

## redirect.toSearchResult(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to redirect a user to a search results page. For example, the results from an on demand search created with the <a href="#">N/search Module</a> , or a loaded search that you modified but did not save.
<b>Returns</b>	Void
<b>Supported Script Types</b>	afterSubmit user event scripts For more information, see the help topic <a href="#">SuiteScript 2.0 User Event Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/redirect Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
options.Search	<a href="#">search.Search</a>	required	

## redirect.toSuitelet(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to redirect the user to a Suitelet.  For more information about Suitelets, see the help topic <a href="#">SuiteScript 2.0 Suitelet Script Type</a> .
 <b>Note:</b>	The redirect happens after the script finishes.
<b>Returns</b>	Void
<b>Supported Script Types</b>	Suitelets, beforeLoad user events, and synchronous afterSubmit user events  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/redirect Module</a>
<b>Since</b>	2015.2

## Parameters

<p><b>Note:</b> The options parameter is a JavaScript object.</p>			
Parameter	Type	Required / Optional	Description
options.scriptId	string	required	The script ID for the Suitelet.
options.deploymentId	string	required	The deployment ID for the Suitelet.
options.isExternal	boolean <code>true</code>   <code>false</code>	optional	The default value is <code>false</code> – indicates an external Suitelet URL.
options.parameters	Object	optional	Contains additional URL parameters as key/value pairs.
<p><b>Note:</b> Parameters cannot be arrays. Use <code>JSON.stringify/JSON.parse</code> instead to handle arrays.</p>			

## Syntax

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

```
//Add additional code
...
redirect.toSuitelet({
 scriptId: 31 ,
 deploymentId: 1,
 parameters: {'custparam_test':'helloWorld'}
});
...
//Add additional code
```

## redirect.toTaskLink(options)

<p><b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.</p>	
<b>Method Description</b>	Method used to redirect a user to a tasklink.
<b>Returns</b>	Void
<b>Supported Script Types</b>	Suitelets, beforeLoad user events, and synchronous afterSubmit user events For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/redirect Module
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description
options.id	string	required	The taskId for a tasklink  For a list of supported task IDs, see the help topic <a href="#">Task IDs</a> .
options.parameters	Object	optional	Contains additional URL parameters as key/value pairs.

## Syntax

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

```
//Add additional code
...
redirect.toTaskLink({
    id: 'ADMI_SHIPPING' ,
    parameters: {'custparam_test':'helloWorld'}
});
...
//Add additional code
```

## N/render Module

**ℹ Note:** The content in this help topic pertains to SuiteScript 2.0.

The render module encapsulates functionality for printing, PDF creation, form creation from templates, and email creation from templates.

**ℹ 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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	render.EmailMergeResult	Object	Server-side scripts	Encapsulates an email merge result
	render.TemplateRenderer	Object	Server-side scripts	Encapsulates a template engine object that produces HTML

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
				and PDF printed forms utilizing advanced PDF/HTML template capabilities
Method	render.bom(options)	file.File	Server-side scripts	Creates a PDF or HTML file object containing a bill of materials
	render.create()	render.Template Renderer	Server-side scripts	Creates a render.TemplateRenderer object
	render.mergeEmail(options)	render.EmailMerge Result	Server-side scripts	Creates a render.EmailMergeResult object
	render.packingSlip(options)	file.File	Server-side scripts	Creates a PDF or HTML file object containing a packing slip
	render.pickingTicket(options)	file.File	Server-side scripts	Creates a PDF or HTML file object containing a picking ticket
	render.statement(options)	file.File	Server-side scripts	Creates a PDF or HTML file object containing a statement
	render.transaction(options)	file.File	Server-side scripts	Creates a PDF or HTML file object containing a transaction
	render.xmlToPdf(options)	file.File	Server-side scripts	Passes XML to the BFO tag library (which is stored by NetSuite), and returns a PDF file
Enum	render.DataSource	enum	Server-side scripts	Holds the string values for supported data source types
	render.PrintMode	enum	Server-side scripts	Holds the string values for supported print output types

## EmailMergeResult Object Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	EmailMergeResult.body	string (read-only)	Server-side scripts	The body of the email distribution in string format
	EmailMergeResult.subject	string (read-only)	Server-side scripts	The subject of the email distribution in string format

## TemplateRenderer Object Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	TemplateRenderer.addCustomDataSource(options)	void	Server-side scripts	Adds an XML file or JSON object to an advanced template as a custom data source
	TemplateRenderer.addQuery(options)	void	Server-side scripts	Uses Query as the renderer's data source
	TemplateRenderer.addRecord(options)	void	Server-side scripts	Binds a record to a template variable

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	TemplateRenderer.addSearchResults(options)	void	Server-side scripts	Binds a search result to a template variable
	TemplateRenderer.renderAsPdf()	Object	Server-side scripts	Uses an advanced template to produce a PDF printed form
	TemplateRenderer.renderPdfToResponse()	void	Server-side scripts	Renders PDF template content as a server response
	TemplateRenderer.renderAsString()	string	Server-side scripts	Returns template content in string form
	TemplateRenderer.setTemplateById(options)	void	Server-side scripts	Sets the template using the internal ID
	TemplateRenderer.setTemplateByscriptId(options)	void	Server-side scripts	Sets the template using the script ID
	TemplateRenderer.renderToResponse(options)	void	Server-side scripts	Renders HTML template content as a server response
Property	TemplateRenderer.templateContent	string	Server-side scripts	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?](#)

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to generate a PDF file from a raw XML string.

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

require(['N/render'],
function(render) {
    function generatePdfFileFromRawXml() {
        var xmlStr = "<?xml version=\"1.0\"?>\n" +
            "<!DOCTYPE pdf PUBLIC "-//big.faceless.org//report\" \"report-1.1.dtd\">\n" +
            "<pdf>\n<body font-size=\"18px\">\nHello World!\n</body>\n</pdf>";
        var pdfFile = render.xmlToPdf({
            xmlString: xmlStr
        });
    }
});
```

```

    }
    generatePdfFileFromRawXml();
});

```

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to render 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.

```

/**
 * @NApiVersion 2.x
 */

require(['N/render'],
  function(render) {
    function renderTransactionToHtml() {
      var transactionFile = render.transaction({
        entityId: 23,
        printMode: render.PrintMode.HTML
      });
    }
    renderTransactionToHtml();
});

```

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to render an invoice into a PDF file using an XML template in the File Cabinet. This sample requires the Advanced PDF/HTML Templates feature.

```

/**
 * @NApiVersion 2.x
 */

// This sample 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 sample, 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.

**Note:** This script sample uses the **define** function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the **require** function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample shows how to render search results into a PDF file.

```

/**
 * @NApiVersion 2.x
 * @NScriptType Suitelet
 */

// This sample 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\" " + "src=\"NetSuiteFonts/verdana.ttf\" " + "src-bold=\"NetSuiteFonts/verdanab.ttf\" " + "src-italic=\"NetSuiteFonts/verdanai.ttf\" " + "src-bolditalic=\"NetSuiteFonts/verdanabi.ttf\" " + "bytes=\"2\"/>\n" + "</head>\n" +
                "<body font-family=\"russianfont\" font-size=\"18\">\n?????? ?????</body>\n" + "</pdf>";

            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({
                templateName: 'exampleName',
                searchResult: results
            });

            var newfile = renderer.renderAsPdf();
            response.writeFile(newfile, false);
        }
    }
)

```

```

        return {
            onRequest: onRequest
        };
    });
}

```

## render.EmailMergeResult

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates an email merge result. Use <a href="#">render.mergeEmail(options)</a> to create and return this object. For a complete list of this object's properties, see <a href="#">EmailMergeResult Object Members</a> .
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/render Module</a>
<b>Since</b>	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',
        id: 62
    },
    recipient: {
        type: 'lead',
        id: 41
    },
    supportCaseId: 2,
    transactionId: 271,
    custmRecord: null
});
...
//Add additional code

```

## EmailMergeResult.body

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The body of the email distribution in string format
-----------------------------	---

Type	string (read-only)
Supported Script Types	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Module	<a href="#">N/render Module</a>
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
...
log.debug({
    title: 'Email Body: ',
    details: mergeResultObj.body
});
...

//Add additional code
```

## EmailMergeResult.subject

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

Property Description	The subject of the email distribution in string format
Type	string (read-only)
Supported Script Types	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Module	<a href="#">N/render Module</a>
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
...
log.debug({
    title: 'Email Subject: ',
    details: mergeResultObj.subject
});
...

//Add additional code
```

## render.TemplateRenderer



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	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 <a href="#">nlobjResponse</a> object, or as XML string that can be passed to <a href="#">render.xmlToPdf(options)</a> 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 <a href="#">TemplateRenderer Object Members</a> .
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/render Module</a>
<b>Since</b>	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();
...
//Add additional code
```

## TemplateRenderer.addCustomDataSource(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds XML or JSON as custom data source to an advanced PDF/HTML template
<b>Returns</b>	Void
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Governance</b>	None
<b>Module</b>	N/render Module
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description
options.alias	string	required	Data source alias
options.format	render.DataSource	required	Data format
options.data	Object   Document   string	required	Object, document, or string

## Syntax

 **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 renderer = render.create();

var xmlObj = xml.Parser.fromString(xmlString);
var jsonObj = JSON.parse(jsonString);

renderer.templateContent = "${XML.book.title}<br />${XML.book.chapter[1].title}<br />${JSON.book.title}<br />
${JSON.book.chapter[1].title}<br />${JSON_STR.book.title}<br />${XML_STR.book.title}";

renderer.addCustomDataSource({
    format: render.DataSource.XML_DOC,
    alias: "XML",
    data: xmlObj
});
renderer.addCustomDataSource({
    format: render.DataSource.XML_STRING,
    alias: "XML_STR",
    data: xmlString
});
renderer.addCustomDataSource({
    format: render.DataSource.OBJECT,
    alias: "JSON",
    data: jsonObj
});
renderer.addCustomDataSource({
    format: render.DataSource.JSON,
    alias: "JSON_STR",
    data: jsonString
});
```

```
var xml = renderer.renderAsString();
...
//Add additional code
```

## TemplateRenderer.addQuery(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Adds a SuiteAnalytics Workbook query to use as the renderer's data source. You can specify the query in two ways:</p> <ul style="list-style-type: none"> <li>▪ As a <a href="#">query.Query</a> object (which you create using the <a href="#">N/query Module</a>) using the <code>options.query</code> parameter.</li> <li>▪ By providing the workbook ID of an existing SuiteAnalytics workbook using the <code>options.id</code> parameter.</li> </ul> <p>One of <code>options.query</code> or <code>options.id</code> is required.</p> <p>This method returns a maximum of 5000 results in the query result set. If a query matches more than 5000 results, you must use <code>options.pageIndex</code> and <code>options.pageSize</code> to retrieve the full set of results.</p> <p>There is no governance for this method. When the renderer is executed, rendering consumes 10 usage units for every iteration through the results. If the query is specified using a workbook ID, rendering consumes an additional 5 usage units before the first iteration through the results.</p>
<b>Returns</b>	Void
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/render Module</a>
<b>Parent Object</b>	<a href="#">render.TemplateRenderer</a>
<b>Sibling Object Members</b>	<a href="#">TemplateRenderer Object Members</a>
<b>Since</b>	2019.2

### Parameters



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

Parameter	Type	Required / Optional	Description
<code>options.id</code>	string	required if <code>options.query</code> is not specified	Workbook query ID.
<code>options.pageIndex</code>	number	optional	Page index.
<code>options.pageSize</code>	number	optional	Page size. The minimum value is 5, and the maximum value is 1000.

Parameter	Type	Required / Optional	Description
options.templateName	string	required	Name of the results iterator variable referred to in the template.
options.query	query.Query object	required if options.id is not specified	Workbook query definition.

## Errors

Error Code	Error Message	Thrown If
MUTUALLY_EXCLUSIVE_ARGUMENTS	Cannot use mutually exclusive arguments.	Both query and ID parameters are provided.
NEITHER_ARGUMENT_DEFINED	One of the following arguments is mandatory: id, query.	Neither options.id nor options.query are provided.
SSS_MISSING_REQD_ARGUMENT	TemplateRenderer.addQuery: Missing a required argument: options.templateName.	The template name is not specified.
UNABLE_TO_LOAD_QUERY	Unable to load query: invalidId.	The query parameter is provided but the ID is not valid.
WRONG_PARAMETER_TYPE	Wrong parameter type: options.query is expected as query.Query.	The query parameter is provided, but the ID has the incorrect type.

## Syntax

Note that `TemplateRenderer.addQuery(options)` binds the results iterator to a template variable. This iterator provides sequential access to the query results. For random access to search results in FreeMarker, process your search data before rendering, and then pass the processed data to the template using `TemplateRenderer.addCustomDataSource(options)`.



**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 search = query.create({
    type: query.Type.TRANSACTION
});
var transactionlines = search.join({
    fieldId: "transactionlines"
});
search.condition = transactionlines.createCondition({
    fieldId: "netamount",
    operator: query.Operator.GREATER,
    values: 1
});
search.columns = [
    transactionlines.createColumn({
        fieldId: "netamount",
        ...
    })
];
```

```

        label: "Amount"
    }),
];

// get instance of TemplateRenderer - https://system.netsuite.com/app/help/helpcenter.nl?fid=section_4412065265.html
var renderer = render.create();
renderer.templateContent = "<#list page as line><item>${line['@label']}:${line[0]}</item>\n</#list>"

renderer.addQuery({
    templateName: "page",
    query: search,
    pageSize: 3, // minimum page size is 5, so result will contain 5 lines instead of 3
    pageIndex: 0,
})
...
// Add additional code

```

## TemplateRenderer.addRecord(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Binds a record to a template variable.
<b>Returns</b>	Void
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/render Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
options.templateName	string	required	Name of the record object variable referred to in the template
options.record	record.Record object	required	The record to add

### Syntax

 **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
    });
});
...
//Add additional code

```

## TemplateRenderer.addSearchResults(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Binds a search result to a template variable.
<b>Returns</b>	Void
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/render Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
options.templateName	string	required	Name of the template
options.searchResult	search.Result object	required	The search result to add

### Syntax

 **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 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({
    templateName: 'exampleName',
    searchResult: results
});
...
//Add additional code

```

## TemplateRenderer.renderAsPdf()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Uses the advanced template to produce a PDF printed form
<b>Returns</b>	<a href="#">file.File</a>
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/render Module</a>
<b>Since</b>	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 invoicePdf = renderer.renderAsPdf();
...
//Add additional code

```

## TemplateRenderer.renderPdfToResponse()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	Void
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None

<b>Module</b>	N/render Module
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description
response	<a href="#">http.ServerResponse</a>	required	Response that will be written to PDF. For example, the response passed from a Suitelet.

## Syntax

 <b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/render Module Script Sample</a> .
---

```
//Add additional code
...
var invoicePdf = renderer.renderPdfToResponse({
    response: myServerResponseObj
});
...
//Add additional code
```

## TemplateRenderer.renderAsString()

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Return template content in string form
<b>Returns</b>	string
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/render Module
<b>Since</b>	2015.2

## Syntax

 <b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/render Module Script Sample</a> .
---

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

```
var invoicePdf = renderer.renderAsString();
...
//Add additional code
```

## TemplateRenderer.renderToResponse(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Writes template content to a server response.
<b>Returns</b>	Void
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/render Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
options.response	<a href="#">http.ServerResponse</a>	required	Response to write to

### Syntax

 **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 invoice = renderer.renderToResponse({
    response: myServerResponseObj
});
...
//Add additional code
```

## TemplateRenderer.setTemplateById(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the template using the internal ID
<b>Returns</b>	Void

<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/render Module</a>
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description
options.id	number	required	Internal ID of the template

## Syntax

 **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 renderer = render.create();
renderer.setTemplateById(3);
var xml = renderer.renderAsString();
...
//Add additional code
```

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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the template using the script ID
<b>Returns</b>	Void
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None

<b>Module</b>	N/render Module
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description
options.scriptId	string	required	Script ID of the template

## Syntax

 **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 renderer = render.create();
renderer.setTemplateByscriptId({
    scriptId: "STDTMPLPRICELIST"
});
var xml = renderer.renderAsString();
...
//Add additional code
```

## TemplateRenderer.templateContent

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Content of template
<b>Type</b>	string
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/render Module</a>
<b>Since</b>	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
...
renderer.templateContent = xmlTemplateFile.getContents();
```

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

## render.bom(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Use this method to create a PDF or HTML object of a bill of material.
<b>Returns</b>	<a href="#">file.File</a> that contains a PDF or HTML document
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	N/render Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
options.entityId	number	required	The internal ID of the bill of material to print
options.printMode	string	optional	The print output type. Set using the <a href="#">render.PrintMode</a> enum.  By default, uses the company/user preference for print output.
options.inCustLocale	boolean <code>true false</code>	optional	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.

### Syntax

 **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 transactionFile = render.bom({
    entityId: 23,
    printMode: render.PrintMode.HTML,
    inCustLocale: true
});
```

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

## render.create()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Use this method to produce HTML and PDF printed forms with advanced PDF/HTML templates.</p> <p>Creates <a href="#">render.TemplateRenderer</a>.</p> <p>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 <a href="#">http.ServerResponse</a>, or as XML string that can be passed to <a href="#">render.xmlToPdf(options)</a> to produce a PDF.</p>
<b>&gt;Returns</b>	<a href="#">render.TemplateRenderer</a>
<b>Supported Script Types</b>	<p>Server-side scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/render Module</a>
<b>Since</b>	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 renderer = render.create();
...
//Add additional code
```

## render.mergeEmail(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a <a href="#">render.EmailMergeResult</a> object for a mail merge with an existing scriptable email template
<b>Returns</b>	<a href="#">render.EmailMergeResult</a>

<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/render Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.templateId	number	required	Internal ID of the template
options.entity	RecordRef	required	Entity
options.recipient	RecordRef	required	Recipient
options.customRecord	RecordRef	required	Custom record
options.supportCaseId	number	required	Support case ID
options.transactionId	number	required	Transaction ID

## 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.

Property	Type	Required / Optional	Description
RecordRef.id	number	required	Internal ID of the record instance
RecordRef.type	string	required	The record type 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/render Module Script Sample](#).

```
//Add additional code
...
var myMergeResult = render.mergeEmail({
    templateId: 1234,
    entity: {
        type: 'employee',
        id: 623
    },
    recipient: {
```

```

        type: 'lead',
        id: 543
    },
supportCaseId: 674,
transactionId: 8987,
customRecord: {
    type: 'custrecord_rewardpoints',
    id: 5423
}
});
...
//Add additional code

```

## render.packingSlip(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Use this method to create a PDF or HTML object of a packing slip.
<b>Returns</b>	<a href="#">file.File</a> that contains a PDF or HTML document
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/render Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
options.entityId	number	required	The internal ID of the packing slip to print
options.printMode	string	optional	The print output type. Set using the <a href="#">render.PrintMode</a> enum.  By default, uses the company/user preference for print output.
options.formId	number	optional	The packing slip form number
options.fulfillmentId	number	optional	Fulfillment ID number
options.inCustLocale	boolean <code>true false</code>	optional	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.

## Syntax

**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 transactionFile = render.packingSlip({
    entityId: 23,
    printMode: render.PrintMode.HTML,
    inCustLocale: true
});
...
//Add additional code
```

## render.pickingTicket(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Use this method to create a PDF or HTML object of a picking ticket.
<b>Returns</b>	<a href="#">file.File</a> that contains a PDF or HTML document
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/render Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.entityId	number	required	The internal ID of the picking ticket to print
options.printMode	string	optional	The print output type. Set using the <a href="#">render.PrintMode</a> enum.  By default, uses the company/user preference for print output.
options.formId	number	optional	The packing slip form number
options.shipgroup	number	optional	Shipping group for the ticket
options.location	number	optional	Location for the ticket
options.inCustLocale	boolean <code>true false</code>	optional	Applies when advanced templates are used. Print the document in the customer's locale.

Parameter	Type	Required / Optional	Description
			If basic printing is used, this parameter is ignored and the transaction form is printed in the customer's locale.

## Syntax

**⚠ 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 transactionFile = render.pickingTicket({
    entityId: 23,
    printMode: render.PrintMode.HTML,
    inCustLocale: true
});
...
//Add additional code
```

## render.statement(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Use this method to create a PDF or HTML object of a statement.
<b>Returns</b>	<a href="#">file.File</a> that contains a PDF or HTML document
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/render Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.entityId	number	required	The internal ID of the statement to print
options.printMode	string	optional	The print output type. Set using the <a href="#">render.PrintMode</a> enum.  By default, uses the company/user preference for print output.
options.formId	number	optional	Internal ID of the form to use to print the statement

Parameter	Type	Required / Optional	Description
options.startDate	Date	optional	Date of the oldest transaction to appear on the statement
options.statementDate	Date	optional	Statement date
options.openTransactionsOnly	boolean true   false	optional	Include only open transactions
options.inCustLocale	boolean true false	optional	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.
options.consolidateStatements	boolean true false	optional	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](#).

```
//Add additional code
...
var transactionFile = render.statement({
    entityId: 23,
    printMode: render.PrintMode.HTML,
    inCustLocale: true
});
...
//Add additional code
```

## render.transaction(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Use this method to create a PDF or HTML object of a transaction.
	 <b>Note:</b> File size is limited to 10MB.
	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>
<b>Returns</b>	file.File that contains a PDF or HTML document
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units

<b>Module</b>	N/render Module
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description
options.entityId	number	required	The internal ID of the transaction to print
options.printMode	enum	optional	The print output type. Set using the <a href="#">render.PrintMode</a> enum.  By default, uses the company/user preference for print output.
options.formId	number	optional	The transaction form number
options.inCustLocale	boolean <code>true false</code>	optional	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.

## Syntax

 <b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/render Module Script Sample</a> .
---

```
//Add additional code
...
var transactionFile = render.transaction({
    entityId: 23,
    printMode: render.PrintMode.HTML,
    inCustLocale: true
});
...
//Add additional code
```

## render.xmlToPdf(options)

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Method used to pass XML to the Big Faceless Organization (BFO) tag library (which is stored by NetSuite), and return a PDF file. BFO is used in NetSuite. For version details, see the help topic <a href="#">Third-Party Notices and Licenses</a> .
 <b>Note:</b>	File size cannot exceed 10MB.

<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/render Module</a>
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description
options.xmlString	<a href="#">xml.Document</a>   string	required	XML document or string to convert to PDF.  For information and examples about using BFO tags to create this document or string, see the help topic <a href="#">nlapiXMLToPDF(xmlstring)</a> , which documents the corresponding SuiteScript 1.0 API.

## Syntax

 <b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/render Module Script Sample</a> .
--

```
//Add additional code
...
var pdfFile = render.xmlToPdf({
    xmlString: xmlStr
});
...
//Add additional code
```

## render.DataSource

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Enum Description</b>	Holds the string values for supported data source types. Use this enum to set the <code>options.format</code> parameter.
	 <b>Note:</b> 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.

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/render Module

## Values

- JSON
- OBJECT
- XML\_DOC
- XML\_STRING

## Syntax

**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
...
renderer.addCustomDataSource({
    format: render.DataSource.JSON,
    alias: "JSON_STR",
    data: jsonString
});
...
//Add additional code
```

## render.PrintMode

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the string values for supported print output types. Use this enum to set the <code>options.printMode</code> parameter.
	<p><b>Note:</b> 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>
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	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](#).

```
//Add additional code
...
printMode: render.PrintMode.HTML
...
//Add additional code
```

# N/runtime Module



**Note:** The content in this help topic pertains to SuiteScript 2.0.

Load the runtime module when you want to view runtime settings for the script, the session, or the user. You can also use this module to set a session key and to see whether a particular feature is enabled in your account.

- [N/runtime Module Members](#)
- [Script Object Members](#)
- [Session Object Members](#)
- [User Object Members](#)
- [N/runtime Module Script Samples](#)

## N/runtime Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">runtime.Script</a>	Object	Client and server scripts	Encapsulates the runtime settings of the currently executing script.
	<a href="#">runtime.Session</a>	Object	Server scripts	Encapsulates the user session for the currently executing script.
	<a href="#">runtime.User</a>	Object	Client and server scripts	Encapsulates the properties and preferences of the user currently executing the script.
Method	<a href="#">runtime.getCurrentScript()</a>	<a href="#">runtime.Script</a>	Client and server scripts	Returns a <a href="#">runtime.Script</a> object that represents the currently executing script.
	<a href="#">runtime.getCurrentSession()</a>	<a href="#">runtime.Session</a>	Client and server scripts	Returns a <a href="#">runtime.Session</a> object that represents the user session for the currently executing script.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	runtime.getCurrentUser()	runtime.User	Client and server scripts	Returns a <a href="#">runtime.User</a> object that represents the properties and preferences of the user currently executing the script.
	runtime.isFeatureInEffect(options)	boolean	Client and server scripts	Indicates whether a particular feature is enabled in a NetSuite account. These are the features that appear on the Enable Features page.
Property	runtime.accountId	string (read-only)	Client and server scripts	The account ID for the current user.
	runtime.envType	string (read-only)	Client and server scripts	The current environment in which the script is executing. This property uses values from the <a href="#">runtime.EnvType</a> enum.
	runtime.executionContext	string (read-only)	Client and server scripts	The trigger of the current script. This property uses values from the <a href="#">runtime.ContextType</a> enum.
	runtime.processorCount	number (read-only)	Client and server scripts	The number of processors available to the current account.
	runtime.queueCount	number (read-only)	Client and server scripts	The number of scheduled script queues available to the current account.
	runtime.version	string (read-only)	Client and server scripts	The version of NetSuite that the method is called in. For example, this property in an account running NetSuite 2015.2 is 2015.2.
Enum	runtime.ContextType	enum	Client and server scripts	Holds the context values for script triggers. This is the type for the <a href="#">runtime.executionContext</a> property.
	runtime.EnvType	enum	Client and server scripts	Holds all possible environment types that the current script can execute in. This is the type for the <a href="#">runtime.envType</a> property.
	runtime.Permission	enum	Client and server scripts	Holds the user permission level for a specific permission ID. This is the type returned by the <a href="#">User.getPermission(options)</a> method.

## Script Object Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Script.getParameter(options)	number   Date   string   boolean	Client and server scripts	Returns the value of a script parameter for the currently executing script.
	Script.getRemainingUsage()	number	Client and server scripts	Returns the number of usage units remaining for the currently executing script.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	Script.apiVersion	string (read-only)	Client and server scripts	The current script's runtime version.
	Script.bundleIds	Array (read-only)	Client and server scripts	An array of bundle IDs for the bundles that include the currently executing script.
	Script.deploymentId	string (read-only)	Server scripts	The deployment ID for the script deployment of the currently executing script.
	Script.id	string (read-only)	Client and server scripts	The script ID for the currently executing script.
	Script.logLevel	string (read-only)	Server scripts	The script logging level for the currently executing script.
	Script.percentComplete	number	Client and server scripts	The percent complete for the current scheduled script execution. This value will appear in the % Complete column on the Scheduled Script Status page.

## Session Object Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Session.get(options)	string   null	Server scripts	Returns the user-defined session object value associated with the session object key. Both the session object value and associated key are defined using <a href="#">Session.set(options)</a> .
	Session.set(options)	void	Server scripts	Sets a key and value for a user-defined session object.

## User Object Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	User.getPermission(options)	string	Client and server scripts	Returns a <a href="#">runtime.Permission</a> user permission level for the specified permission.
	User.getPreference(options)	string	Client and server scripts	Returns the value of a NetSuite preference.  Currently only General Preferences and Accounting Preferences are exposed in SuiteScript. For more information about these preferences, see the help topics <a href="#">General Preferences</a> and <a href="#">Accounting Preferences</a> .

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	User.contact	number(read-only)	Client and server scripts	The internal ID of the currently logged-in contact.
	User.department	number (read-only)	Client and server scripts	The internal ID of the department for the current user.
	User.email	string (read-only)	Client and server scripts	The email address of the current user.
	User.id	number (read-only)	Client and server scripts	The internal ID of the current user.
	User.location	number (read-only)	Client and server scripts	The internal ID of the location of the current user.
	User.name	string (read-only)	Client and server scripts	The name of the current user.
	User.role	number (read-only)	Client and server scripts	The internal ID of the role for the current user.
	User.roleCenter	string (read-only)	Client and server scripts	The string value of the center type, or role center, for the current user.
	User.roleId	string (read-only)	Client and server scripts	The custom scriptId of the role for the current user.
	User.subsidiary	number (read-only)	Client and server scripts	The internal ID of the subsidiary for the current user.

## N/runtime Module Script Samples

**i Note:** This script sample uses the **define** function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the **require** function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample shows how to use a Suitelet to write user and session information for the currently executing script to the response.

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

//Write user and session information for the currently executing script to the response.
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
};
});

```

**Note:** This script sample uses the `define` function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the `require` function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample shows how to use a scheduled script to create multiple sales records and logs the record creation progress.

```

/**
 * @NApiVersion 2.x
 * @NScriptType ScheduledScript
 */

//Create multiple sales records and log the record creation progress.
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
                });
                script.percentComplete = (x * 100)/500;
                log.debug({
                    title: 'New Sales Orders',
                    details: "Record creation progress: " + script.percentComplete + "%"
                });
            }
        }
    });
});

```

## runtimrue.Script

**Note:** The content in this help topic pertains to SuiteScript 2.0.

### Object Description

Encapsulates the runtime settings of the currently executing script.

	<p>Use <code>runtime.getCurrentScript()</code> to access this object.</p> <p>For a complete list of this object's methods and properties, see <a href="#">Script Object Members</a>.</p>
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript(); //scriptObj is a runtime.Script object
log.debug('Script ID: ' + scriptObj.id);
...
//Add additional code
```

## Script.getParameter(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the value of a script parameter for the currently executing script.  For information on script parameters, see the help topic <a href="#">Creating Script Parameters Overview</a> .
<b>Returns</b>	number   Date   string   boolean
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	2015.2

## Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	The name of the script parameter.	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 Samples](#).

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
log.debug("Script parameter of custscript1: " +
    scriptObj.getParameter({name: 'custscript1'}));
...
//Add additional code
```

## Script.getRemainingUsage()

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the number of usage units remaining for the currently executing script. For more information, see the help topic <a href="#">SuiteScript Governance</a> .
<b>Returns</b>	number
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/runtime Module
<b>Since</b>	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/runtime Module Script Samples](#).

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
log.debug("Remaining governance units: " + scriptObj.getRemainingUsage());
...
//Add additional code
```

## Script.apiVersion

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The current script's runtime version.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Module</b>	N/runtime Module
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
var scriptApiVersion = scriptObj.apiVersion;
...
//Add additional code
```

## Script.bundleIds



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	An array of bundle IDs for the bundles that include the currently executing script.  When you use this property, consider the following: <ul style="list-style-type: none"><li>■ The array can contain any number of bundle IDs, because a bundle installation script can be associated with multiple bundles.</li><li>■ The array does not contain bundle IDs of deprecated bundles.</li><li>■ The elements in the array are sorted in ascending order from the lowest bundle ID to the highest bundle ID that includes the currently executing script.  However, it is not guaranteed that the last element in the array corresponds to the bundle that triggered the currently executing script during the bundle installation.</li></ul>
<b>Type</b>	string [] (read-only)
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
var bundleArr = scriptObj.bundleIds;
...
//Add additional code
```

## Script.deploymentId

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The deployment ID for the script deployment on the currently executing script.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/runtime Module
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
log.debug("Deployment Id: " + scriptObj.deploymentId);
...
//Add additional code
```

## Script.id

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The script ID for the currently executing script.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/runtime Module
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
log.debug("Script Id: " + scriptObj.id);
...
```

```
//Add additional code
```

## Script.logLevel

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The script logging level for the currently executing script. This method is not supported on client scripts.  Returns one of the following values: <ul style="list-style-type: none"><li>■ DEBUG</li><li>■ AUDIT</li><li>■ ERROR</li><li>■ EMERGENCY</li></ul>
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
log.debug("Logging level: " + scriptObj.logLevel);
...
//Add additional code
```

## Script.percentComplete

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The percent complete specified for the current scheduled script execution. This value appears in the % Complete column on the Scheduled Script Status page.  This value can be set or retrieved.
<b>Type</b>	number
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
SSS_OPERATION_UNAVAILABLE		The currently executing script is not a scheduled script.

## Syntax

**Important:** The following code snippets show the syntax for this member. They are not a functional examples. For a complete script example, see [N/runtime Module Script Samples](#).

```
//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
```

## runtim.Session

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	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 <a href="#">Session.set(options)</a> to set session object values. Use <a href="#">Session.get(options)</a> to retrieve session object values.
---------------------------	--

	For a complete list of this object's methods, see <a href="#">Session Object Members</a> .
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var sessionObj = runtime.getCurrentSession(); //sessionObj is a runtime.Session object
sessionObj.set({
    name: "myKey",
    value: "myValue"
});
log.debug("Session object myKey value: " + sessionObj.get({name: "myKey"}));
...
//Add additional code
```

## Session.get(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the user-defined session object value associated with a session object key.  Both the session object value and associated key are defined using <a href="#">Session.set(options)</a> .  If the key does not exist, this method returns null.
<b>Returns</b>	string   null
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	Key used to store the session object.	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 Samples](#).

```
//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)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets a key and value for a user-defined session object.  Use <a href="#">Session.get(options)</a> to retrieve the object value after you set it.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	Key used to store the session object.	2015.2
options.value	string	required	Value to associate with the key in the user session.	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 Samples](#).

```
//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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the properties and preferences of the user currently executing the script.  For a complete list of this object's methods and properties, see <a href="#">User Object Members</a> .
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var userObj = runtime.getCurrentUser(); //sessionObj is a runtime.User object
...
//Add additional code
```

## User.getPermission(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a <a href="#">runtime.Permission</a> user permission level for the specified permission.
<b>Returns</b>	string
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/runtime Module</a>

<b>Since</b>	2015.2
--------------	--------

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	Internal ID of a permission. For a list of permission IDs, see <a href="#">Permission Names and IDs</a> .	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 Samples](#).

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
var userPermission = userObj.getPermission ({
    name = 'ADMI_ACCOUNTING'
});
log.debug("User permission of ADMI_ACCOUNTING:" +
    (userPermission ==
        runtime.Permission.FULL?'FULL':userPermission));
...
//Add additional code
```

## User.getPreference(options)

<b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.
--

<b>Method Description</b>	Returns the value set for a NetSuite preference.  Currently only General Preferences and Accounting Preferences are exposed in SuiteScript.  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 <code>config.load(options)</code> , where <code>options.name</code> is <code>COMPANY_PREFERENCES</code> or <code>ACCOUNTING_PREFERENCES</code> . The <code>config.load(options)</code> method returns a <code>Record</code> . You can use the <code>Record.setValue(options)</code> method to set the preference.  For more information about these preferences, see the help topics <a href="#">General Preferences</a> and <a href="#">Accounting Preferences</a> .
<b>Returns</b>	string

<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	Internal ID of the preference. For a list of preference IDs, see <a href="#">Permission Names and IDs</a> .	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 Samples](#).

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
var userPref = userObj.getPreference ({
    name: "emailemployeeonapproval"
});
log.debug("User preference for emailemployeeonapproval: " + userPref);
...
//Add additional code
```

## User.contact

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal ID of the currently logged-in contact.  If no logged-in entity or other entity than contact is logged in, then <b>0</b> is returned as value.
<b>Type</b>	number (read-only)
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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/runtime Module Script Samples](#).

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current contact: " + userObj.contact);
...
//Add additional code
```

## User.department

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal ID of the department for the current user.
<b>Type</b>	number (read-only)
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/runtime Module
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user department: " + userObj.department);
...
//Add additional code
```

## User.email

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The email address of the current user. To use this property, the <b>email</b> field on the user employee record must contain an email address.
-----------------------------	---

	<p><b>Note:</b> 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.</p>
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	<p>Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Current user email: " + userObj.email);
...
//Add additional code
```

## User.id

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal ID of the current user.
<b>Type</b>	number (read-only)
<b>Supported Script Types</b>	<p>Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user: " + userObj.id);
...
//Add additional code
```

## User.location



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal ID of the location of the current user.
<b>Type</b>	number (read-only)
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user location: " + userObj.location);
...
//Add additional code
```

## User.name



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The name of the current user.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
```

```
log.debug("Name of current user: " + userObj.name);
...
//Add additional code
```

## User.role



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal ID of the role for the current user.
<b>Type</b>	number (read-only)
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user role: " + userObj.role);
...
//Add additional code
```

## User.roleCenter



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The string value of the center type, or role center, for the current user.  The NetSuite UI adjusts automatically to different users' business needs. For each user, NetSuite displays a variable set of tabbed pages, called a <b>center</b> , 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 <a href="#">Centers Overview</a> .
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("String value of current user center type (role center): " + userObj.roleCenter);
...
//Add additional code
```

## User.roleId



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The custom scriptId of the role for the current user.  You can use this value instead of <code>User.role</code> . 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 access the unique/custom scriptId assigned to the role.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Custom script ID of current user role: " + userObj.roleId);
...
//Add additional code
```

## User.subsidiary



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal ID of the subsidiary for the current user.
-----------------------------	---

Type	number (read-only)
Supported Script Types	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Module	<a href="#">N/runtime Module</a>
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 Samples](#).

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user subsidiary: " + userObj.subsidiary);
...
//Add additional code
```

## runtime.getCurrentScript()

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

Method Description	Returns a <a href="#">runtime.Script</a> object 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 <a href="#">runtime.getCurrentSession()</a> or <a href="#">runtime.getCurrentUser()</a> instead.
Returns	<a href="#">runtime.Script</a>
Supported Script Types	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Governance	None
Module	<a href="#">N/runtime Module</a>
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 Samples](#).

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
...
```

```
//Add additional code
```

## runtime.getCurrentSession()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a <a href="#">runtime.Session</a> object that represents the user session for 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 user, use <a href="#">runtime.getCurrentScript()</a> or <a href="#">runtime.getCurrentUser()</a> instead.
<b>Returns</b>	<a href="#">runtime.Session</a>
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var sessionObj = runtime.getCurrentSession();
...
//Add additional code
```

## runtime.getCurrentUser()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a <a href="#">runtime.User</a> object that represents the properties and preferences for the user currently executing the 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 <a href="#">runtime.getCurrentScript()</a> or <a href="#">runtime.getCurrentSession()</a> instead.
<b>Returns</b>	<a href="#">runtime.User</a>
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None

<b>Module</b>	N/runtime Module
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
...
//Add additional code
```

## runtime.isFeatureInEffect(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 > Company > Enable Features.
<b>Returns</b>	boolean
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/runtime Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.feature	string	required	The internal ID of the feature to check. For a list of feature internal IDs, see the help topic <a href="#">Feature Names and IDs</a> .	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 Samples](#).

```
//Add additional code
```

```

...
var featureInEffect = runtime.isFeatureInEffect ({
    feature: "ADVBILLING"});
log.debug('Advanced Billing feature is enabled: ' + featureInEffect);
...
//Add additional code

```

## runtime.accountId



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The account ID for the current user.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```

//Add additional code
...
log.debug("Account ID for the current user: " + runtime.accountId);
...
//Add additional code

```

## runtime.envType



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The current environment in which the script is executing. This property uses values from the <a href="#">runtime.EnvType</a> enum.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
log.debug("Environment for current user: " + JSON.stringify(runtime.envType));
...
//Add additional code
```

## runtime.executionContext

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The trigger of the current script. This property uses values from the <a href="#">runtime.ContextType</a> enum.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
if (runtime.executionContext === runtime.ContextType.USEREVENT)
    return;
...
//Add additional code
```

## runtime.processorCount

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The number of processors available to the current account.
-----------------------------	--

	<p><a href="#">SuiteCloud Processors</a> 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.</p> <p>For scheduled script deployments that continue to use queues, use <a href="#">runtime.queueCount</a>. 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 <a href="#">SuiteCloud Processors – Supported Task Types</a>).</p> <p>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 <a href="#">SuiteCloud Plus Settings</a>.</p> <p><b>Note:</b> The <code>runtime.processorCount</code> 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 <a href="#">SuiteCloud Processors – Supported Task Types</a>.</p>
<b>Type</b>	number (read-only)
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/runtime Module
<b>Since</b>	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/runtime Module Script Samples](#).

```
//Add additional code
...
log.debug("Number of processors available: " + runtime.processorCount);
...
//Add additional code
```

## runtime.queueCount

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>The number of scheduled script queues available to the current account.</p> <p><a href="#">SuiteCloud Processors</a> 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.</p> <p>For map/reduce script deployments, use <a href="#">runtime.processorCount</a>. With the introduction of SuiteCloud Processors, no map/reduce script deployments use queues (see the help topic <a href="#">SuiteCloud Processors – Supported Task Types</a>).</p> <p>Be aware that the number of queues available may not be the same as the number of processors available (see the help topic <a href="#">SuiteCloud Plus Settings</a>).</p>
-----------------------------	--

	<p><b>Note:</b> If all scheduled script deployments in an account are configured to no longer use queues (see the help topic <a href="#">SuiteCloud Processors – Supported Task Types</a>), the value of <code>runtime.queueCount</code> is unchanged. This property reflects the number of queues available to an account. It is not impacted by changes to deployments.</p>
	For more information on scheduled scripts, see the help topic <a href="#">SuiteScript 2.0 Scheduled Script Type</a> .
<b>Type</b>	number (read-only)
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
...
log.debug("Number of queues available: " + runtime.queueCount);
...
//Add additional code
```

## runtime.version

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The version of NetSuite that the method is called in. For example, the <code>runtime.version</code> property in an account running NetSuite 2015.2 is 2015.2.  For example, you can use this method when installing a bundle in another NetSuite accounts to find out the version number before installing the bundle.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	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 Samples](#).

```
//Add additional code
```

```

...
log.debug("Current NetSuite version: " + runtime.version);
...
//Add additional code

```

## runtime.ContextType

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the context values for script triggers.  This is the type for the <a href="#">runtime.executionContext</a> property.
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	2015.2

### Values

Enum Value	Sets runtime.ExecutionContext Property To
ACTION	ACTION
BUNDLE_INSTALLATION	BUNDLEINSTALLATION
CLIENT	CLIENT
CONSOLRATEADJUSTOR	CONSOLRATEADJUSTOR
CSV_IMPORT	CSVIMPORT
CUSTOMGLLINES	CUSTOMGLLINES
CUSTOM_MASSUPDATE	CUSTOMMASSUPDATE
DEBUGGER	DEBUGGER
EMAIL_CAPTURE	EMAILCAPTURE
MAP_REDUCE	MAPREDUCE
NONE	NONE
PAYMENTGATEWAY	PAYMENTGATEWAY
PORTLET	PORTLET
PROMOTIONS	PROMOTIONS

Enum Value	Sets runtime.ExecutionContext Property To
RESTLET	RESTLET
REST_WEBSERVICES	RESTWEBSERVICES
SCHEDULED	SCHEDULED
SDF_INSTALLATION	SDFINSTALLATION
SHIPPING_PARTNERS	SHIPPINGPARTNERS
SUITELET	SUITELET
TAX_CALCULATION	TAXCALCULATION
USEREVENT	USEREVENT
USER_INTERFACE	USERINTERFACE
WEBAPPLICATION	WEBAPPLICATION
SOAP_WEBSERVICES	SOAPWEBSERVICES
WEBSTORE	WEBSTORE
WORKFLOW	WORKFLOW

## runtime.EnvType



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	<p>Holds all possible environment types that the current script can execute in. This is the type for the <a href="#">runtime.envType</a> property.</p> <div style="border: 1px solid #ccc; padding: 5px;"> <p><b>Note:</b> 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> </div>
<b>Supported Script Types</b>	<p>Client and server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	2015.2

### Values

<ul style="list-style-type: none"> <li>■ SANDBOX</li> <li>■ PRODUCTION</li> <li>■ BETA</li> <li>■ INTERNAL</li> </ul>
---

## runtime.Permission



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the user permission level for a specific permission ID. This is the type returned by the <a href="#">User.getPermission(options)</a> method.  For information on working with NetSuite permissions, see the help topics <a href="#">NetSuite Permissions Overview</a> and <a href="#">Permission Names and IDs</a> .
<b>Supported Script Types</b>	Client and server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/runtime Module</a>
<b>Since</b>	2015.2

### Values

- FULL
- EDIT
- CREATE
- VIEW
- NONE

## N/search Module



**Note:** The content in this help topic pertains to SuiteScript 2.0.

Load the search module to create and run on-demand or saved searches and analyze and iterate through the search results. You can use this module to do the following:

- Search for a single record using keywords
- Create and save searches
- Load and run previously saved searches
- Search for duplicate records
- Return a set of records that match filter criteria you define

You can also 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.

### In this help topic

- [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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">search.Search</a>	Object	Client and server-side scripts	Encapsulates a NetSuite search. Use the methods available to the Search object to create a search, run a search, or save a search.
	<a href="#">search.Result</a>	Object	Client and server-side scripts	Encapsulate a single search result row. Use the methods and properties for the Result object to get the column values for the result row.
	<a href="#">search.Column</a>	Object	Client and server-side scripts	Encapsulates a single search column in a <a href="#">search.Search</a> object. Use the methods and properties available to the Column object to get or set Column properties.
	<a href="#">search.Filter</a>	Object	Client and server-side scripts	Encapsulates a search filter used in a search. Use the properties for the Filter object to get and set the filter properties.
	<a href="#">search.ResultSet</a>	Object	Client and server-side scripts	Encapsulates a set of search results returned by <a href="#">Search.run()</a> .
	<a href="#">search.Page</a>	Object	Client and server-side scripts	Encapsulates a set of search results for a single search page.
	<a href="#">search.PagedData</a>	Object	Client and server-side scripts	Holds metadata about a paginated query.
	<a href="#">search.PageRange</a>	Object	Client and server-side scripts	Defines the page range to bound the result set for a paginated query.
	<a href="#">search.Settings</a>	Object	Client and server-side scripts	Encapsulates a search setting. Search settings let you specify search parameters that are typically available only in the UI.
Method	<a href="#">search.create(options)</a>	<a href="#">search.Search</a>	Client and server-side scripts	Creates a new search and returns it as a <a href="#">search.Search</a> object.
	<a href="#">search.create.promise(options)</a>	<a href="#">search.Search</a>	Client scripts	Creates a new search asynchronously and returns it as a <a href="#">search.Search</a> object.
	<a href="#">search.load(options)</a>	<a href="#">search.Search</a>	Client and server-side scripts	Loads an existing saved search and returns it as a <a href="#">search.Search</a> object.
	<a href="#">search.load.promise(options)</a>	<a href="#">search.Search</a>	Client scripts	Loads an existing saved search asynchronously and returns it as a <a href="#">search.Search</a> object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	search.delete(options)	void	Client and server-side scripts	Deletes an existing saved search asynchronously and returns it as a <a href="#">search.Search</a> object.
	search.delete.promise(options)	void	Client scripts	Deletes an existing saved search and returns it as a <a href="#">search.Search</a> object.
	search.duplicates(options)	<a href="#">search.Result</a> []	Client and server-side scripts	Performs a search for duplicate records based on the duplicate detection configuration for the account. Returns an array of <a href="#">search.Result</a> objects.
	search.duplicates.promise(options)	<a href="#">search.Result</a> []	Client scripts	Performs a search for duplicate records asynchronously based on the duplicate detection configuration for the account. Returns an array of <a href="#">search.Result</a> objects.
	search.global(options)	<a href="#">search.Result</a> []	Client and server-side scripts	Performs a global search against a single keyword or multiple keywords.
	search.global.promise(options)	<a href="#">search.Result</a> []	Client scripts	Performs a global search asynchronously against a single keyword or multiple keywords.
	search.lookupFields(options)	Object   array	Client and server-side scripts	Performs a search 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.
	search.lookupFields.promise(options)	Object   array	Client scripts	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.
	search.createColumn(options)	<a href="#">search.Column</a>	Client and server-side scripts	Creates a new search column as a <a href="#">search.Column</a> object.
	search.createFilter(options)	<a href="#">search.Filter</a>	Client and server-side scripts	Creates a new search filter as a <a href="#">search.Filter</a> object.
	search.createSetting(options)	<a href="#">search.Settings</a>	Client and server-side scripts	Creates a new search setting and returns it as a <a href="#">search.Settings</a> object.
Enum	search.Operator	enum	Client and server-side scripts	Enumeration that holds the values for search operators to use with the <a href="#">search.Filter</a> object.
	search.Sort	enum	Client and server-side scripts	Enumeration that holds the values for supported sorting directions used with <a href="#">search.createColumn(options)</a> .
	search.Summary	enum	Client and server-side scripts	Enumeration that holds the values for summary types used by the <a href="#">Column.summary</a> object.
	search.Type	enum	Client and server-side scripts	Enumeration that holds the string values for search types supported in the <a href="#">N/search Module</a> . This enum is used to pass the type argument to <a href="#">search.create(options)</a> .

## Search Object Members

The following members are called on [search.Search](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Search.save()	number	Client and server-side scripts	Saves a search created by <a href="#">search.create(options)</a> or loaded with <a href="#">search.load(options)</a> . Returns the internal ID of the saved search.
	Search.save.promise()	number	Client scripts	Asynchronously saves a search created by <a href="#">search.create(options)</a> or loaded with <a href="#">search.load(options)</a> . Returns the internal ID of the saved search.
	Search.run()	search.ResultSet	Client and server-side scripts	Runs an on demand search created with <a href="#">search.create(options)</a> or a search loaded with <a href="#">search.load(options)</a> , returning the results as a <a href="#">search.ResultSet</a> .
	Search.runPaged(options)	search.PagedData	Client and server-side scripts	Runs the current search and returns a <a href="#">search.PagedData</a> Object.
	Search.runPaged.promise(options)	search.PagedData	Client scripts	Asynchronously runs the current search and returns a <a href="#">search.PagedData</a> Object.
Property	Search.searchType	string	Client and server-side scripts	Search type on which a search is based.
	Search.searchId	number	Client and server-side scripts	Internal ID of a search.
	Search.filters	search.Filter[]	Client and server-side scripts	Filters for the search as an array of <a href="#">search.Filter</a> objects.
	Search.filterExpression	Object[]	Client and server-side scripts	Search filter expression for the search as an array of expression objects.
	Search.columns	search.Column[]   string[]	Client and server-side scripts	Columns to return for this search as an array of <a href="#">search.Column</a> objects or a string array of column names.
	Search.packageId	string	Client and server-side scripts	The application ID for the search.
	Search.settings	search.Setting[]   string[]	Client and server-side scripts	Search settings for this search as an array of <a href="#">search.Setting</a> objects or a string array of column names.
	Search.title	string	Client and server-side scripts	Title for a saved search. Use this property to set the title for a search before you save it for the first time.
	Search.id	string	Client and server-side scripts	Script ID for a saved search, starting with <a href="#">customsearch</a> .
	Search.isPublic	boolean true   false	Client and server-side scripts	Value is <b>true</b> if the search is public, or <b>false</b> if it is not.

## Column Object Members

The following members are called on [search.Column](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Column.setWhenOrderedBy(options)	search.Column	Client and server-side scripts	Returns the search column for which the minimal or maximal

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
				value should be found when returning the <a href="#">search.Column</a> value.
Property	<a href="#">Column.name</a>	string (read-only)	Client and server-side scripts	Name of a search column as a string.
	<a href="#">Column.join</a>	string (read-only)	Client and server-side scripts	Join ID for a search column as a string.
	<a href="#">Column.summary</a>	string (read-only)	Client and server-side scripts	Returns the summary type for a search column.
	<a href="#">Column.formula</a>	string	Client and server-side scripts	Formula used for a search column as a string.
	<a href="#">Column.label</a>	string	Client and server-side scripts	Label used for the search column. You can only get or set custom labels with this property.
	<a href="#">Column.function</a>	string	Client and server-side scripts	Special function used in the search column as a string.

## Filter Object Members

The following members are called on [search.Filter](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	<a href="#">Filter.name</a>	string (read-only)	Client and server-side scripts	Name or internal ID of the search field.
	<a href="#">Filter.join</a>	string (read-only)	Client and server-side scripts	Join ID for the search filter.
	<a href="#">Filter.operator</a>	string (read-only)	Client and server-side scripts	Operator used for the search filter.
	<a href="#">Filter.summary</a>	<a href="#">search.Summary</a>	Client and server-side scripts	Summary type for the search filter.
	<a href="#">Filter.formula</a>	string	Client and server-side scripts	Formula used by the search filter.

## Page Object Members

The following members are called on the [search.Page](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">Page.next()</a>	void	Client and server-side scripts	Gets the next segment of data from a paginated search

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	<a href="#">Page.next.promise()</a>	void	Client scripts	Asynchronously gets the next segment of data from a paginated search
	<a href="#">Page.prev()</a>	void	Client and server-side scripts	Gets the previous segment of data from a paginated search
	<a href="#">Page.prev.promise()</a>	void	Client scripts	Asynchronously gets the previous segment of data from a paginated search
Property	<a href="#">Page.data</a>	<a href="#">search.Result[]</a>	Client and server-side scripts	The results from a paginated search.
	<a href="#">Page.isFirst</a>	read-only boolean	Client and server-side scripts	Indicates whether a page is the first page of data for a result set
	<a href="#">Page.isLast</a>	read-only boolean	Client and server-side scripts	Indicates whether a page is the last page of data for a result set.
	<a href="#">Page.pagedData</a>	read-only <a href="#">search.PagedData</a>	Client and server-side scripts	The <b>PagedData</b> Object used to fetch this <b>Page</b> Object.
	<a href="#">Page.pageRange</a>	read-only <a href="#">search.PageRange</a>	Client and server-side scripts	The <b>PageRange</b> Object used to fetch this <b>Page</b> Object.

## PagedData Object Members

The following members are called on [search.PagedData](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">PagedData.fetch(options)</a>	<a href="#">search.Page</a>	Client and server-side scripts	Retrieves the data within the specified page range.
	<a href="#">PagedData.fetch.promise()</a>	<a href="#">search.Page</a>	Client scripts	Asynchronously retrieves the data within the specified page range.
Property	<a href="#">PagedData.count</a>	read-only number	Client and server-side scripts	The total number of results when <a href="#">Search.runPaged(options)</a> was executed.
	<a href="#">PagedData.pageRanges</a>	read-only <a href="#">search.PageRange[]</a>	Client and server-side scripts	The collection of <b>PageRange</b> objects that divide the entire result set into smaller groups.
	<a href="#">PagedData.pageSize</a>	read-only number	Client and server-side scripts	The maximum number of entries per page
	<a href="#">PagedData.searchDefinition</a>	read-only <a href="#">search.Search</a>	Client and server-side scripts	The search criteria used when <a href="#">Search.runPaged(options)</a> was executed.

## PageRange Object Members

The following members are called on [search.PageRange](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	<a href="#">PageRange.compoundLabel</a>	read-only string	Client and server-side scripts	Human-readable label with beginning and ending range identifiers
	<a href="#">PageRange.index</a>	read-only number	Client and server-side scripts	The index of this page range.

## Result Object Members

The following members are called on [search.Result](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">Result.getValue(options)</a>	string	Client and server-side scripts	Used on formula fields and non-formula (standard) fields to get the value of a specified search return column.
	<a href="#">Result.getValue(column)</a>	string	Client and server-side scripts	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 <a href="#">search.Column</a> Object.
	<a href="#">Result.getText(column)</a>	string	Client and server-side scripts	The text value for a <a href="#">search.Column</a> if it is a stored select field.
	<a href="#">Result.getText(options)</a>	string	Client and server-side scripts	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.
Property	<a href="#">Result.recordType</a>	string (read-only)	Client and server-side scripts	The type of record returned in a search result row.
	<a href="#">Result.id</a>	string (read-only)	Client and server-side scripts	The internal ID for the record returned in a search result row.
	<a href="#">Result.columns</a>	<a href="#">search.Column</a> []	Client and server-side scripts	Array of <a href="#">search.Column</a> objects that encapsulate the columns returned in the search result row.

## ResultSet Object Members

The following members are called on [search.ResultSet](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">ResultSet.getRange(options)</a>	<a href="#">search.Result</a> []	Client and server-side scripts	Retrieve a slice of the search result as an array of <a href="#">search.Result</a> objects.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	ResultSet.getRange.promise(options)	search.Result[]	Client scripts	Asynchronously retrieve a slice of the search result as an array of <code>search.Result</code> objects.
	ResultSet.each(callback)	void	Client and server-side scripts	Use a developer-defined function to invoke on each row in the search results, up to 4000 results at a time.
	ResultSet.each.promise(callback)	void	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 <code>search.Column</code> objects that represent the columns returned in the search results.

## Setting Object Members

The following members are called on `search.Settings`.

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

The following script samples demonstrate how to use the features of the N/search module.



**Important:** These samples are designed to run in a NetSuite OneWorld account, so the OneWorld feature must be enabled in your NetSuite account for the samples to work.

### Sample 1: Create and run a search



**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample creates a search for customer records. The sample specifies several result columns and one filter, and it logs the first 50 search results.

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

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

    var myResultSet = mySearch.run();

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

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

## Sample 2: Create and save a search

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample creates a search for sales order records and saves it. The sample specifies several result columns and two filters.

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

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']
            ]
        });
        mySalesOrderSearch.save();
    }

    createSearch();
});
```

## Sample 3: Load and run a search

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample loads and runs a saved search for sales order records. The sample uses the `each` callback function to process the results.

```
/*
 * @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'
            });

            return true;
        });
    }

    loadAndRunSearch();
});
}
```

## Sample 4: Run a search and get a range of result rows

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample loads and runs a saved search for sales order records. The sample obtains the first 100 rows of search results.

```
/*
 * @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();
});

```

## Sample 5: Run a paginated search

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample loads and runs a saved search for sales order records. The sample uses the `forEach` callback function to process the paginated results.

```

/**
 * @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();

```

```
});
```

## Sample 6: Search for a custom record type

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample creates a search for a custom record type. To search for a custom record type, you must specify a type of `search.Type.CUSTOM_RECORD` and add the ID of the custom record type (as a string). In this sample, the ID of the custom record type is 6. The custom record also includes a custom field named `custrecord1`.

```
/**  
 * @NApiVersion 2.x  
 */  
  
require(['N/search'], function(search) {  
    var myCustomRecordSearch = search.create({  
        type: search.Type.CUSTOM_RECORD + '6';  
        title: 'My Search Title',  
        columns: ['custrecord1']  
    }).run().each(function(result) {  
        // Process each result  
        return true;  
    });  
});
```

## Sample 7: Search in a custom list

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample creates a search for items in a custom list. It searches for the internal ID value of an abbreviation in a custom list named `customlist_mylist`.

```
/**  
 * @NApiVersion 2.x  
 */  
  
require(['N/search'], function(search) {  
    var internalId = -1;  
    var myCustomListSearch = search.create({  
        type: 'customlist_mylist',  
        columns: [  
            { name : 'internalId' },  
            { name : 'abbreviation' }  
        ]  
    });  
});
```

```

myCustomListSearch.filters = [
    search.createFilter({
        name: 'formulatext',
        formula: '{abbreviation}',
        operator: search.Operator.IS,
        values: abbreviation
    });
]

var resultSet = myCustomListSearch.run();
var results = resultSet.getRange({
    start: 0,
    end: 1
});
for(var i in results) {
    // log.debug('Found custom list record', results[i]);
    internalId = results[i].getValue({
        name:'internalId'
    });
}
);
}
);

```

## Sample 8: Delete a saved search

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to delete a saved search.

```

/**
 * @NApiVersion 2.x
 */

require(['N/search'], function(search) {
    function deleteSearch() {
        search.delete({
            id: 'customsearch_my_so_search'
        });
    }

    deleteSearch();
});

```

## search.Search

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a NetSuite search. Use the methods available to <code>search.Search</code> to create a search, run a search, or save a search.
---------------------------	---

	<p> <b>Note:</b> You do not need to save the search to run it.</p> <p>For a complete list of this object's methods and properties, see <a href="#">Search Object Members</a>.</p>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});
...
//Add additional code
```

## Search.run()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Runs an on-demand search created with <a href="#">search.create(options)</a> or a search loaded with <a href="#">search.load(options)</a>, returning the results as a <a href="#">search.ResultSet</a>. Calling this method does not save the search.</p> <p>Use this method with <a href="#">search.create(options)</a> to create and run on-demand searches that are never saved to the database.</p> <p>After you run a search, you can use <a href="#">ResultSet.each(callback)</a> to iterate through the result set and process each result.</p> <p> <b>Important:</b> When you call this method, consider the following:</p> <ul style="list-style-type: none"> <li>▪ 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.</li> <li>▪ For better performance, consider creating a saved search in the UI and loading it in your script using <a href="#">search.load(options)</a> instead of creating the search directly in your script using <a href="#">search.create(options)</a>.</li> </ul>
<b>Returns</b>	<a href="#">search.ResultSet</a>
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/search Module</a>

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
...
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;
    });
}

```

## Search.runPaged(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Runs the current search and returns summary information about paginated results.</p> <p>Calling this method does not give you the result set or save the search.</p> <p>To retrieve data, use <a href="#">PagedData.fetch(options)</a>.</p> <p><b>Important:</b> When you use this method to run a paged search, consider the following:</p> <ul style="list-style-type: none"> <li>▪ 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.</li> <li>▪ This method can return a maximum of 1000 pages of search results.</li> </ul>
<b>Returns</b>	<a href="#">search.PagedData</a>
<b>Supported Script Types</b>	<p>All script types</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2016.1

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description
options.pageSize	number	optional	<p>Maximum number of entries per page</p> <p>There is an upper limit, a lower limit, and a default setting:</p> <ul style="list-style-type: none"> <li>■ The maximum number allowed is 1000.</li> <li>■ The minimum number allowed is 5.</li> <li>■ By default, the page size is set to 50 entries per page.</li> </ul>

## Syntax

 <b>Important:</b>	The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/search Module Script Samples</a> .
---	--

```
//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)

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Runs the current search asynchronously and returns a <a href="#">search.PagedData</a> Object.
 <b>Note:</b>	The parameters and errors thrown for this method are the same as those for <a href="#">Search.runPaged(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .

<b>Synchronous Version</b>	<code>Search.runPaged(options)</code>
<b>Supported Script Types</b>	All client-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2016.1

## Syntax

 **Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
//Add additional code
...
mySearch.runPaged.promise().then(getPageRangesPromiseChain);
...
//Add additional code
```

## Search.save()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Saves a search created by <a href="#">search.create(options)</a> or loaded with <a href="#">search.load(options)</a> . Returns the internal ID of the saved search.  You must set the title and id properties for a new saved search before you save it, either when you create it with <a href="#">search.create(options)</a> or by setting the <code>Search.title</code> and <code>Search.id</code> properties.  If you do not set the saved search ID, NetSuite generates one for you. See <a href="#">Search.id</a> .   <b>Note:</b> You do not need to set these properties if you load a previously saved search with <a href="#">search.load(options)</a> and then save it.  This method also includes a promise version, <code>Search.save.promise()</code> . For more information about promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	the internal search ID of the saved search as a number
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	{1}: Missing a required argument: {2}	Required <code>Search.title</code> property not set on <code>search.Search</code> .
NAME_ALREADY_IN_USE	A search has already been saved with that name. Please use a different name.	The <code>Search.title</code> property on <code>search.Search</code> is not unique.
SSS_DUPLICATE_SEARCH_SCRIPT_ID	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> .	The <code>Search.id</code> property on <code>search.Search</code> is not unique.

## Syntax



**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
...
mySalesOrderSearch.save();
...
//Add additional code
```

## Search.save.promise()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Asynchronously 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.
	<span style="border: 1px solid #0070C0; padding: 2px 10px; border-radius: 10px;"><b>Note:</b> The parameters and errors thrown for this method are the same as those for <code>Search.save()</code>. For more information on promises, see <a href="#">Promise Object</a>.</span>
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<code>Search.save()</code>
<b>Supported Script Types</b>	All client-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

## Syntax

**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
//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

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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 <code>customer</code> ; if the search was on the Sales Order record type, this property is <code>salesorder</code> .
<b>Type</b>	read-only string
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
```

```

    });
log.debug({
    title: 'record type: ',
    details: mySearch.searchType
});
...
//Add additional code

```

## Search.searchId



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Internal ID of the search.  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.
<b>Type</b>	number
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});
log.debug({
    title: 'search id #: ',
    details: mySearch.searchId
});
...
//Add additional code

```

## Search.filters



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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.
-----------------------------	--

	You set this value with an array or single search.Filter 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.
	<p><b>Note:</b> If you want to get or set a search filter expression, use the <code>Search.filterExpression</code> property.</p>
<b>Type</b>	<code>search.Filter[]</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/search Module
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
SSS_INVALID_SRCH_FILTER	An search filter contains invalid search criteria	Invalid value for search filter 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/search Module Script Samples](#).

```
//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

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Use filter expressions as a shortcut to create filters ( <code>search.Filter</code> ).
-----------------------------	--

	<p>A search filter expression is a JavaScript string array of zero or more elements. Each element is one of the following:</p> <ul style="list-style-type: none"> <li>■ Operator - For a list of supported operators, see <a href="#">search.Operator</a>.</li> <li>■ Filter term</li> <li>■ Two or more filter expressions combined logically with 'and', 'or', or 'not'</li> </ul> <p>Use <code>null</code> to set an empty array and remove any existing filter expressions on this search.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <span style="color: #0070C0; font-size: 1.5em; border-radius: 50%; padding: 2px 5px; margin-right: 5px;"></span> <b>Note:</b> If you want to get or set search filters, use the <a href="#">Search.filters</a> property.         </div>
<b>Type</b>	Object[]
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
SSS_INVALID_SRCH_FILTER_EXPR	Malformed search filter expression. This is a general error raised when a filter expression cannot be parsed. For example: <code>[ f1, 'and', 'and', f2 ]</code>	The <code>options.filters</code> 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](#).

```
//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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Columns to return for this search as an array of <code>search.Column</code> objects or a string array of column names.  You set this value with an array of <code>search.Column</code> objects or a single <code>search.Column</code> to overwrite any prior return columns for the search. Use <code>null</code> to set an empty array and remove any existing columns on this search.
<b>Type</b>	<code>search.Column[]   string[]</code>
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

### Errors

Error Code	Thrown If
<code>SSS_INVALID_SRCH_COLUMN</code>	The value passed in was not a string or <code>search.Column</code> 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/search Module Script Samples](#).

```
//Add additional code
...
function createSearch() {
    var mySalesOrderSearch = search.create({
        type: search.Type.SALES_ORDER,
        columns: ['entity', 'subsidiary', 'name', 'currency'],
    });
...
//Add additional code
```

## Search.packageId

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The application ID for this search.  An application ID identifies a SuiteApp project and is a fully qualified name with the following notation:
-----------------------------	---

	<p>&lt;publisher_id&gt;.&lt;project_id&gt;</p> <p>For example, com.netsuite.mysuiteapp and org.mycompany.helloworld are application IDs.</p> <p>To use this feature, the Show App ID Field preference must be enabled in your NetSuite account. For more information, see the help topic <a href="#">SDF Account Preferences (SDF Developers Only)</a>.</p>
<b>Type</b>	string
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2019.2

## 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
...
var mySalesOrderSearch = search.create({
    type: search.Type.SALES_ORDER,
    packageId: 'com.example',
    columns: ['entity', 'subsidiary', 'name', 'currency']
});

var thePackageId = mySalesOrderSearch.packageId;
...
// Add additional code
```

## Search.settings

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>Search settings for this search as an array of <a href="#">search.Setting</a> objects or a string array of column names. Search settings let you specify search parameters that are typically available only in the UI.</p> <p>You set this value with an array of <a href="#">search.Setting</a> objects or a single <a href="#">search.Setting</a> object. You can create a <a href="#">search.Setting</a> object by calling <a href="#">search.createSetting(options)</a>. You can also set this value with an array of column names, each of which is a string.</p> <p>The supported values for a <a href="#">search.Setting</a> object differ depending on the search parameter that you set. For more information, see <a href="#">Setting.name</a> and <a href="#">Setting.value</a>.</p>
<b>Type</b>	<a href="#">search.Setting[]</a>   string[]
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>

Since	2018.2
-------	--------

## Errors

Error Code	Thrown If
SSS_INVALID_SRCH_SETTING	An unknown search parameter name is provided.
SSS_INVALID_SRCH_SETTING_VALUE	An unsupported value is set for the provided search parameter name.

## Syntax

 **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 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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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 <a href="#">search.create(options)</a> .  The Search.title property is required to save a search with <a href="#">Search.save()</a> .
<b>Type</b>	string
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>

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
...
function createSearch() {
    var mySalesOrderSearch = search.create({
        type: search.Type.SALES_ORDER,
        title: 'My SalesOrder Search',
        id: 'customsearch_my_so_search',
    });
    mySalesOrderSearch.save();
...
//Add additional code
```

## Search.id

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Script ID for a saved search, starting with <code>customsearch</code> . If you do not set this property and then save the search, NetSuite generates a script ID for you.  <b>Note:</b> This is not the internal NetSuite ID for the saved search. See <a href="#">Search.searchId</a> .
<b>Type</b>	number
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Value is <code>true</code> if the search is public, or <code>false</code> if it is not. By default, all searches created through <code>search.create(options)</code> are private.
<b>Type</b>	<code>boolean true   false</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});
mySearch.isPublic = true;
...
//Add additional code

```

## search.Result



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulate a single search result row. Use the methods and properties for <code>search.Result</code> to get the column values for the result row.  <div style="background-color: #e0f2ff; padding: 5px;"> <b>Note:</b> Use <code>search.ResultSet</code> for the set of results from a search.       </div>
	For more information about executing NetSuite searches using SuiteScript, see <a href="#">Searching Overview</a> . For a complete list of this object's methods and properties, see <a href="#">Result Object Members</a> ,

<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
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'
    });
}
...
//Add additional code
```

## Result.getValue(column)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Used on formula and non-formula (standard) fields. Returns the value of a specified search result column. For convenience, this method takes a single <a href="#">search.Column</a> Object.
<b>Returns</b>	<p>The return type depends on the type of search result column that was specified:</p> <ul style="list-style-type: none"> <li>■ boolean if the column is a check box field</li> <li>■ number if the column is a record, list, decimal number, or image field, with the following considerations:           <ul style="list-style-type: none"> <li>□ For image fields, the returned number represents the ID of the image file.</li> </ul> </li> <li>■ string for all other column types, with the following considerations:           <ul style="list-style-type: none"> <li>□ For multiselect fields, the returned string represents a comma-separated list of IDs. Each ID represents a selectable option in the field.</li> </ul> </li> </ul>

	<ul style="list-style-type: none"> <li>For date/time fields, the returned string represents the formatted string value of the date. You can use methods in the N/format module to work with this string (for example, converting it to a Date object). For more information, see <a href="#">N/format Module</a>.</li> </ul>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

## Parameters

Parameter	Type	Required / Optional	Description
column	<a href="#">search.Column</a>	Required	The search result column from which to return a 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/search Module Script Samples](#).

```
//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:',
    details: value
});
...
//Add additional code
```

## Result.getValue(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Used on formula and non-formula (standard) fields. Returns the value of a specified search result column. Takes in arguments for name, join, and summary.
---------------------------	---

	<p><b>Note:</b> This method is overloaded. You can also use <code>Result.getValue(column)</code> to get column values. This method takes in a single <code>search.Column</code>.</p> <p><b>Important:</b> If you have multiple search return columns and you apply grouping, all columns must include a summary property.</p>
<b>Returns</b>	The return type depends on the type of search result column that was specified: <ul style="list-style-type: none"> <li>■ boolean if the column is a check box field</li> <li>■ number if the column is a record, list, decimal number, or image field, with the following considerations: <ul style="list-style-type: none"> <li>□ For image fields, the returned number represents the ID of the image file.</li> </ul> </li> <li>■ string for all other column types, with the following considerations: <ul style="list-style-type: none"> <li>□ For multiselect fields, the returned string represents a comma-separated list of IDs. Each ID represents a selectable option in the field.</li> <li>□ For date/time fields, the returned string represents the formatted string value of the date. You can use methods in the N/format module to work with this string (for example, converting it to a Date object). For more information, see <a href="#">N/format Module</a>.</li> </ul> </li> </ul>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description
<code>options.name</code>	string	Required	The search return column name.
<code>options.join</code>	string	Optional	The join id for this search return column.  Join IDs are listed in the Records Browser. For more information, see the help topic <a href="#">Working with the SuiteScript Records Browser</a> .
<code>options.summary</code>	<a href="#">search.Summary</a>	Optional	The summary type for this column. See <a href="#">search.Summary</a> .
<code>options.func</code>	string	Optional	Special function for the search column. See <a href="#">Column.function</a> .

## Syntax

<b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/search Module Script Samples</a> .
--

```
//Add additional code
```

```

...
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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Used on select, image, and document fields. Returns the text value of a specified search result column. For convenience, this method takes a single <a href="#">search.Column</a> Object.
	<span style="border: 1px solid #0070C0; border-radius: 50%; padding: 2px 5px; margin-right: 5px;"></span> <b>Note:</b> This method is overloaded. You can also use <a href="#">Result.getText(options)</a> to get column text value based on the name, join and summary values for a column.
<b>Returns</b>	string
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

### Parameters

Parameter	Type	Required / Optional	Description
column	<a href="#">search.Column</a>	Required	Name of the search result column.

## Syntax



**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 mySearch = search.load({
  id: 'customsearch_my_so_search'
});

var resultSet = mySearch.run();
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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Used on select, image, and document fields. Returns the text value of a specified search result column.
	 <b>Note:</b> This method is overloaded. You can also use <a href="#">Result.getText(column)</a> to get a column value. This method takes in a single <a href="#">search.Column</a> .
<b>Returns</b>	string
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
options.name	string	Required	The name of the search column.
options.join	string	Optional	The join internal ID for the search column.

Parameter	Type	Required / Optional	Description
options.summary	search.Summary	Optional	The summary type used for the search column. See <a href="#">search.Summary</a> .

## Syntax



**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 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The type of record returned in a search result row.
<b>Type</b>	search.Type enum
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
```

```

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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal ID for the record returned in a search result row. This ID is a number, but it is stored in this property as a string (for example, '237').
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
var mySearch = search.create({
    type: search.Type.CUSTOMER
});

var resultSet = mySearch.run();

resultSet.each(function(result) {
    log.debug({
        title: 'Record Internal ID: ',
        details: result.id
    });

    return true;
});
...
// Add additional code

```

## Result.columns



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Array of <a href="#">search.Column</a> objects that encapsulate the columns returned in the search result row.
<b>Type</b>	<a href="#">search.Column[]</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a single search column in a <a href="#">search.Search</a> . Use the methods and properties available to the Column object to get or set Column properties.
---------------------------	---

	<p>You create a search column object with <code>search.createColumn(options)</code> and add it to a <code>search.Search</code> object that you create with <code>search.create(options)</code> or load with <code>search.load(options)</code>.</p> <p>You can pass a Column object as a parameter to the <code>Result.getValue(column)</code> or <code>Result.getText(column)</code> methods.</p> <p>In addition, <code>search.ResultSet</code> contains an array of Column objects returned in the results of a search.</p> <p>For a complete list of this object's methods and properties, see <a href="#">Column Object Members</a>.</p>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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',
        'entity',
        'entity.firstname',
        'entity.email',
        search.createColumn({
            name: 'formulatext',
            formula: "{lastname}||', '||{firstname}"
        })
    ],
    ...
    // When the search is executed, the corresponding column in the result will then contain a value in the
    form: Last Name, First Name
    ...
//Add additional code
```

## Column.setWhenOrderedBy(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
---------------------------	---

	<b>Note:</b> You can only use this method if you use <b>MIN</b> or <b>MAX</b> as the summary type on a search column with the <a href="#">Result.getValue(options)</a> method.
<b>Returns</b>	<code>search.Column</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description
<code>options.name</code>	string	Required	The name of the search column for which the minimal or maximal value should be found.
<code>options.join</code>	string	Required	The join id for the search column.

## Syntax

<b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/search Module Script Samples</a> .
--

```
//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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Name of a search column as a string.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
// 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

```

## Column.join



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Join ID for a search column as a string.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
// Create a joined column. The name parameter is a field name on
// the joined record type, and the join parameter is a field name
// on the base record type that references the joined record type.
//
// In this example, a search is created for customer records.
// Customer records include a 'salesrep' field, which references an
// employee record. This joined column represents the value of the
// 'firstname' field on the referenced employee record.
var myColumn = search.createColumn({
    name: 'firstname',
    join: 'salesrep'
});

var mySearch = search.create({
    type: search.Type.CUSTOMER,
    columns: [
        'entityid',
        'email',

```

```

        myColumn
    ]
});

var theJoin = myColumn.join;
...
// Add additional code

```

## Column.summary



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns the summary type for a search column.
<b>Type</b>	<a href="#">search.Summary</a> enum
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
log.debug({
    details: 'Summary Type for Search Column: '
        + columnObj.summary
});
...
//Add additional code

```

## Column.formula



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Formula used for a search column as a string. To set this value, you must use formulatext, formulanumeric, formuladatetime, formulapercent, or formulacurrency.
<b>Type</b>	string
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Module</b>	N/search Module
<b>Since</b>	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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Label used for the search column. You can only get or set custom labels with this property.
<b>Type</b>	string
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
var columnObj = search.createColumn({
  name: 'formulanumeric',
  label: 'Numeric Formula'
});
...

```

```
//Add additional code
```

## Column.function

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Special function applied to values in a search column. See <a href="#">Supported Functions</a> .
<b>Type</b>	string
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

### Supported Functions

The following table lists the supported functions and their internal IDs:

Internal ID	Name	Date Function	Output
percentOfTotal	% of Total	No	percent
absoluteValue	Absolute Value	No	integer
ageInDays	Age In Days	Yes	integer
ageInHours	Age In Hours	Yes	integer
ageInMonths	Age In Months	Yes	integer
ageInWeeks	Age In Weeks	Yes	integer
ageInYears	Age In Years	Yes	integer
calendarWeek	Calendar Week	Yes	date
day	Day	Yes	date
month	Month	Yes	text
negate	Negate	No	integer
numberAsTime	Number as Time	No	text
quarter	Quarter	Yes	text
rank	Rank	No	integer
round	Round	No	float
roundToHundredths	Round to Hundredths	No	float
roundToTenths	Round to Tenths	No	float
weekOfYear	Week of Year	Yes	text
year	Year	Yes	text

## Errors

Error Code	Message	Thrown If
INVALID_SRCH_FUNCTN	A search.Column contains an invalid function: {1}.	Unknown function is 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/search Module Script Samples](#).

```
//Add additional code
...
var columnObj = search.createColumn({ // the age of the sales order in days
    name: 'trandate',
    function: 'ageInDays'
});
...
//Add additional code
```

## Column.sort

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The sort order of the column.  Use <a href="#">search.createColumn(options)</a> and a value from the <a href="#">search.Sort</a> enum to set the value of this property. If <code>Column.sort</code> is not set, the column is not sorted in any particular order.  After you create a column, you cannot change the sort order of the column. If you use the same column in another search and specify a new sort order, the previous sort order is still used.
<b>Type</b>	<a href="#">search.Sort</a> enum
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
var columnObj = search.createColumn({
    name: 'invoice',
    sort: search.Sort.DESC
});
...
...
```

```
//Add additional code
```

## search.Filter



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	<p>Encapsulates a search filter used in a search. Use the properties for the Filter object to get and set the filter properties.</p> <p>You create a search filter object with <a href="#">search.createFilter(options)</a> and add it to a <a href="#">search.Search</a> object that you create with <a href="#">search.create(options)</a> or load with <a href="#">search.load(options)</a>.</p> <p><b>Note:</b> NetSuite uses an implicit AND operator with search filters, as opposed to filter expressions which explicitly use either AND and OR operators.</p> <p>Use the following guidelines with the Filter object:</p> <ul style="list-style-type: none"> <li>■ 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 <a href="#">Filter.formula</a> property.</li> <li>■ To search on checkbox fields, use the IS operator with a value of T or F to search for checked or unchecked fields, respectively.</li> </ul> <p>For a complete list of this object's methods and properties, see <a href="#">Filter Object Members</a>.</p>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
var mySearchFilter = search.createFilter({
    name: 'entity',
    operator: search.Operator.ISEMPTY,
});
...
//Add additional code
```

## Filter.name



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Name or internal ID of the search field as a string. For more information, see <a href="#">search.createFilter(options)</a> .
-----------------------------	--

Type	string (read-only)
Supported Script Types	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Module	<a href="#">N/search Module</a>
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
...
log.debug({
    details: 'Filter Name: '
        + filterObj.name
});
...
//Add additional code
```

## Filter.join

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

Property Description	Join ID for the search filter as a string.
Type	string (read-only)
Supported Script Types	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Module	<a href="#">N/search Module</a>
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
...
// 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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Operator used for the search filter. This value is set with the <a href="#">search.Operator</a> enum. The <a href="#">search.Operator</a> enum contains the valid operator values for this property.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
var mySearchFilter = search.createFilter({
    name: 'entity',
    operator: search.Operator.ISEMPTY
});
log.debug({
    details: 'Operator Used: '
        + mySearchFilter.operator
});
...
//Add additional code

```

## Filter.summary

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Summary type for the search filter. Use this property to get or set the value of the summary type. See <a href="#">search.Summary</a> .
<b>Type</b>	<a href="#">search.Summary</a>
<b>Supported Script Types</b>	All script types

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
SSS_INVALID_SRCH_FILTER_SUM	A search.Filter contains an invalid summary type: {1}.	Unknown summary type is 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/search Module Script Samples](#).

```
//Add additional code
...
var mySearchFilter = search.createFilter({
    name: 'entity',
    operator: search.Operator.ISNOTEMPTY,
    summary: search.Summary.GROUP
});
...
//Add additional code
```

## Filter.formula

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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 <a href="#">search.createFilter(options)</a> .
<b>Type</b>	string
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
```

```

...
log.debug({
    details: 'Search Filter Formula: '
        + filterObj.formula
});
...
//Add additional code

```

## search.ResultSet



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	<p>Encapsulates a set of search results returned by <a href="#">Search.run()</a>.</p> <p>Use the methods and properties for the <b>ResultSet</b> object to iterate through each result returned by the search or access an arbitrary slice of results. The maximum number of results in a <b>ResultSet</b> object is 4000. If a search matches more than 4000 results, you must use <a href="#">Search.runPaged(options)</a> or <a href="#">Search.runPaged.promise(options)</a> to retrieve the full set of results.</p> <p>For a complete list of this object's methods and properties, see <a href="#">ResultSet Object Members</a>.</p> <div style="border: 1px solid #f0e68c; padding: 5px; margin-top: 10px;"> <span style="color: yellow;"><b>!</b></span> <b>Important:</b> 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.       </div>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
// 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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Retrieve a slice of the search result as an array of <a href="#">search.Result</a> objects.</p> <p>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.</p> <p>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 <a href="#">search.Result</a> objects.</p> <p>If you specify a range for which there are no results, an empty array is returned. For example, if there are 25 search results, then getRange(30, 40) will return an empty array.</p>
<b>Returns</b>	<a href="#">search.Result[]</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.start</code>	number	Required	Index number of the first result to return, inclusive.
<code>options.end</code>	number	Required	Index number of the last result to return, exclusive.

## Syntax

**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 results = rs.getRange({
    start: 0,
    end: 1000
});
...
//Add additional code
```

## ResultSet.getRange.promise(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to asynchronously retrieve a slice of the search result as an array of <a href="#">search.Result</a> objects.
	<p><b>Note:</b> For information about the parameters and errors thrown for this method, see <a href="#">ResultSet.getRange(options)</a>. For additional information on promises, see <a href="#">Promise Object</a>.</p>
<b>Returns</b>	<a href="#">search.Result[]</a>
<b>Supported Script Types</b>	All client-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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](#).

```
//Add additional code
...
var results = rs.getRange.promise({
    start: 0,
    end: 1000
})
.then(function(response){
    log.debug({
        title: 'Completed',
        details: response
    });
})
```

```

    .catch(function onRejected(reason) {
        log.debug({
            title: 'Failed: ',
            details: reason
        });
    })
    ...
}

//Add additional code

```

## ResultSet.each(callback)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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:  <code>boolean callback(result.Result result);</code>
	The callback function takes a <a href="#">search.Result</a> object as an input parameter and returns a boolean which can be used to stop the iteration with a value of <code>false</code> , or continue the iteration with a value of <code>true</code> .
	<p> <b>Important:</b> 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.</p>
<b>Returns</b>	<code>void</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
callback	function	Required	Named JavaScript function or anonymous inline function that contains the logic to process a <a href="#">search.Result</a> 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/search Module Script Samples](#).

```
//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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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:  <code>boolean callback(result.Result result);</code>
	<div style="background-color: #e0f2ff; padding: 10px;"> <span style="color: #0070C0; font-size: 1.5em;">i</span> <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">ResultSet.each(callback)</a>. For more information on promises, see <a href="#">Promise Object</a>.       </div>
<b>Returns</b>	Promise Object
<b>Supported Script Types</b>	All client-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

### Syntax



**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```

//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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	An array of <code>search.Column</code> objects that represent the columns returned in the search results.
<b>Type</b>	<code>search.Column[]</code> This property is read-only
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	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 <a href="#">Page Object Members</a> .
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to fetch the next segment of data (bounded by <a href="#">search.PageRange</a> ).  Moves the current page to next range.
<b>Returns</b>	Void
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2016.1

## Errors

Error Code	Message	Thrown If
INVALID_PAGE_RANGE	Invalid page range.	The page range is invalid, or when the page is the last page.

## Syntax

 **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.next.promise()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to asynchronously fetch the next segment of data (bounded by <a href="#">search.PageRange</a> ).  Moves the current page to another range. The promise is complete when the data for this range is loaded or rejected.   <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">Page.next()</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">Page.next()</a>
<b>Supported Script Types</b>	All client-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2016.1

## Syntax

 **Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
// 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()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to fetch the previous segment of data (bounded by <code>search.PageRange</code> ).  Moves the current page to previous range.
<b>Returns</b>	Void
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2016.1

### Errors

Error Code	Message	Thrown If
INVALID_PAGE_RANGE	Invalid page range.	The page range is invalid, or when the page is the first page.

### Syntax



**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.isLast){
    page = page.prev();
}
//Add additional code
```

## Page.prev.promise()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to asynchronously fetch the previous segment of data (bounded by <code>search.PageRange</code> ).  Moves the current page to another range. The promise is complete when the data for this range is loaded or rejected.
---------------------------	---

	<b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">Page.prev()</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">Page.prev()</a>
<b>Supported Script Types</b>	All client-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	5 units
<b>Module</b>	N/search Module
<b>Since</b>	2016.1

## Syntax

**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
//Add additional code
...
return mypage.prev.promise().then(processPage);
...
//Add additional code
```

## Page.data

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The results from a paginated search.
<b>Type</b>	<a href="#">search.Result[]</a> This property is read-only.
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/search Module
<b>Since</b>	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
...
function processPage(page){
    page.data.forEach(function(value){
        log.debug({
```

```

        details: "data: " + page.data
    });
...
//Add additional code

```

## Page.isFirst

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether the page is within the first range of the result set. Flags the start of the data collection.
<b>Type</b>	boolean true   false  This property is read-only.
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether a page is within the last range of the result set. Flags the end of the data collection.
<b>Type</b>	boolean true   false  This property is read-only.
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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.isLast){
    page = page.prev();
}
//Add additional code
```

## Page.pagedData

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The PagedData Object used to fetch this Page Object.
<b>Type</b>	<code>search.PagedData</code> This property is read-only.
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
var lastPageRange = pagedData.pageRanges[pagedData.pageRanges.length - 1];
...
//Add additional code
```

## Page.pageRange

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The PageRange Object used to fetch this Page Object.  Page boundary information with the key and label.
<b>Type</b>	<code>search.PageRange</code> This property is read-only.
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Module</b>	N/search Module
<b>Since</b>	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
...
log.debug({
  details: "Page Range: " + mySearchPage.pageRange
});
...

//Add additional code
```

## search.PagedData

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	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 paged searches, the maximum number of result rows per page is 1000. The minimum number of result rows per page is 5, except for the last page in the result set (because the last page may include fewer than 5 results).  For a complete list of this object's methods and properties, see <a href="#">PagedData Object Members</a> .
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/search Module
<b>Since</b>	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
...
// 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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">Promise Object</a> .
<b>Returns</b>	<a href="#">search.Page</a>
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2016.1

### Parameters

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

Parameter	Type	Required / Optional	Description
pageRange.index	number	required	The index of the page range that bounds the desired data.

### Errors

Error Code	Message	Thrown If
INVALID_PAGE_RANGE	Invalid page range.	The page range is not valid.

### Syntax

 **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
```

## PagedData.fetch.promise()

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	This method asynchronously retrieves the data bounded by the <code>pageRange</code> parameter.
	<b>i Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">PagedData.fetch(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">PagedData.fetch(options)</a>
<b>Supported Script Types</b>	All client-side scripts  For more information see, <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2016.1

### Syntax

**⚠ Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
//Add additional code
...
return pagedData.fetch.promise().then(processPage);
...
//Add additional code
```

## PagedData.count

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The total number of results when <a href="#">Search.runPaged(options)</a> was executed.
<b>Type</b>	number  This property is read-only.
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
log.debug({
    details: "Result Count: " + myPagedData.count
});
...
//Add additional code
```

## PagedData.pageRanges

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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.
<b>Type</b>	<a href="#">search.PageRange[]</a>  This property is read-only.
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
var mySearch = search.create({
    type: search.Type.CUSTOMER,
    columns: [
        'entityid',
        'email'
    ]
});

var myPagedResults = mySearch.runPaged({
    pageSize: 10
});

var thePageRanges = myPagedResults.pageRanges;
```

```
// Use PagedData.fetch(options) to retrieve a page of results (as a
// search.Page object) by index
var theData = myPagedResults.fetch({
  index: 5
});
...
// Add additional code
```

## PagedData.pageSize



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Maximum number of entries per page Possible values are 5 - 1000 entries per page.
<b>Type</b>	number This is a read-only property.
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
var mySearch = search.create({
  type: search.Type.CUSTOMER,
  columns: [
    'entityid',
    'email'
  ]
});

var myPagedResults = mySearch.runPaged({
  pageSize: 10
});

// In this example, the value of myPagedResults.pageSize is 10
var thePageSize = myPagedResults.pageSize;

// Use PagedData.fetch(options) to retrieve a page of results (as a
// search.Page object) by index
var theData = myPagedResults.fetch({
  index: 5
});
```

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

## PagedData.searchDefinition

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The search criteria used to execute the result set for this PagedData Object.
<b>Type</b>	read-only <a href="#">search.Search</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
log.debug({
    details: "Search Details: " + myPagedData.searchDefinition
});
...
//Add additional code
```

## search.PageRange

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Defines the page range to contain the result set For a complete list of this object's properties, see <a href="#">PageRange Object Members</a> .
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
});
...
//Add additional code
```

## PageRange.compoundLabel

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Human-readable label with beginning and ending range identifiers
<b>Type</b>	read-only string
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
log.debug({
    details: "Page Range Description: " + myPageRange.compoundLabel
});
...
//Add additional code
```

## PageRange.index

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The index of the pageRange
<b>Type</b>	number This property is read-only.
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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
...
log.debug({
    details: "Page Range Index: " + myPageRange.index
});
...
//Add additional code
```

## search.setting



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	<p>Defines a search setting.</p> <p>Search settings let you specify search parameters that are typically available only in the UI. The following settings are supported:</p> <ul style="list-style-type: none"> <li>■ <b>Consolidated Exchange Rate:</b> 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.</li> <li>■ <b>Show Period End Transactions:</b> 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.</li> </ul> <p>Use <a href="#">search.createSetting(options)</a> to create a setting. After you create your settings, assign them as array values to <a href="#">Search.settings</a>.</p> <p>For a complete list of this object's properties, see <a href="#">Setting Object Members</a>.</p>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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](#).

```
//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

```

## Setting.name



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The name of the search parameter.  This property is set when you call <code>search.createSetting(options)</code> . The following values are supported for this property: <ul style="list-style-type: none"><li>■ <code>consolidationtype</code>: This value corresponds to the Consolidated Exchange Rate setting.</li><li>■ <code>includeperiodendtransactions</code>: This value corresponds to the Show Period End Transactions setting.</li></ul>
<b>Type</b>	string  This property is read-only.
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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](#).

```

//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

```

## Setting.value



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>The value of the search parameter.</p> <p>This property is set when you call <code>search.createSetting(options)</code>. If you specify <code>consolidationtype</code> as the search parameter name (<code>Setting.name</code>), the following values are supported for this parameter:</p> <ul style="list-style-type: none"> <li>■ ACCTTYPE</li> <li>■ AVERAGE</li> <li>■ CURRENT</li> <li>■ HISTORICAL</li> <li>■ NONE</li> </ul> <p>If you specify <code>includeperiodendtransactions</code> as the search parameter name (<code>Setting.name</code>), the following values are supported for this parameter:</p> <ul style="list-style-type: none"> <li>■ F</li> <li>■ FALSE</li> <li>■ T</li> <li>■ TRUE</li> </ul> <p>These values are not case sensitive.</p>
<b>Type</b>	string This property is read-only.
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	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](#).

```

//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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates a new search and returns it as a <a href="#">search.Search</a> object.</p> <p>The search can be modified and run as an on demand search with <a href="#">Search.run()</a>, without saving it. Alternatively, calling <a href="#">Search.save()</a> will save the search to the database, so it can be reused later in the UI or loaded with <a href="#">search.load(options)</a>.</p>
	<p><b>Note:</b> 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.</p> <p>The <code>search.create(options)</code> method also includes a promise version, <code>search.create.promise(options)</code>. For more information about promises, see <a href="#">Promise Object</a>.</p>
	<p><b>Important:</b> When you use this method to create a search, consider the following:</p> <ul style="list-style-type: none"> <li>■ 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.</li> <li>■ 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>.</li> </ul>
<b>Returns</b>	<code>search.Search</code>
<b>Supported Script Types</b>	<p>All script types</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Governance</b>	None
<b>Module</b>	N/search Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.type	string	Required	The search type that you want to base the search on. Use the <a href="#">search.Type</a> enum for this argument.	2015.2
options.filters	<a href="#">search.Filter[]</a>   Object[]	Optional	<p>A single <a href="#">search.Filter</a> object, an array of <a href="#">search.Filter</a> objects, a search filter expression, or an array of search filter expressions.</p> <p>A search filter expression can be passed in as an Object with the following properties:</p> <ul style="list-style-type: none"> <li>■ name (required)</li> <li>■ join</li> <li>■ operator (required)</li> <li>■ summary</li> <li>■ formula</li> <li>■ values</li> </ul> <p>For more information about these properties, see <a href="#">Filter Object Members</a>.</p> <p>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.</p> <p><b>Note:</b> You can further filter the returned <a href="#">search.Search</a> object by adding additional filters with <a href="#">Search.filters</a> or <a href="#">Search.filterExpression</a>.</p>	2015.2
options.filterExpression	Object[]	Optional	<p>Search filter expression for the search as an array of expression objects.</p> <p>A search filter expression is a JavaScript string array of zero or more elements. Each element is one of the following:</p> <ul style="list-style-type: none"> <li>■ Operator - either 'NOT', 'AND', or 'OR'</li> <li>■ Filter term</li> <li>■ Nested search filter expression</li> </ul> <p>You set this value with an array of expression objects or single filter expression object to overwrite any prior filter expressions. Use <code>null</code> to set an empty array and remove any existing filter expressions on this search.</p> <p><b>Note:</b> If you want to get or set a search filters, use the <a href="#">Search.filters</a> property.</p> <p>This parameter sets the value for the <a href="#">Search.filterExpression</a> property.</p>	2016.2
options.columns	<a href="#">search.Column[]</a>   Object[]	Optional	A single <a href="#">search.Column</a> object or array of <a href="#">search.Column</a> objects.	2015.2

Parameter	Type	Required / Optional	Description	Since
			<p>You can optionally pass in an Object or array of Objects with the following properties to represent a Column:</p> <ul style="list-style-type: none"> <li>▪ name (required)</li> <li>▪ formula</li> <li>▪ function</li> <li>▪ join</li> <li>▪ label</li> <li>▪ sort</li> <li>▪ summary</li> </ul> <p>For more information about these properties, see <a href="#">Column Object Members</a>.</p>	
<code>options.packageId</code>	string	Optional	The application ID for this search.	2019.2
<code>options.settings</code>	<code>search. Setting[]   Object[]</code>	Optional	<p>Search settings for this search as a single <code>search.Settings</code> object or an array of <code>search.Settings</code> objects. Search settings let you specify search parameters that are typically available only in the UI. See <a href="#">Search.settings</a>.</p> <p>You can optionally pass in an Object or array of Objects with the following properties to represent a setting:</p> <ul style="list-style-type: none"> <li>▪ name</li> <li>▪ value</li> </ul> <p>For more information about these properties, see <a href="#">Setting Object Members</a>.</p>	2018.2
<code>options.title</code>	string	Optional	The name for a saved search. The title property is required to save a search with <a href="#">Search.save()</a> .	2015.2
<code>options.id</code>	string	Optional	Script ID for a saved search. If you do not set the saved search ID, NetSuite generates one for you. See <a href="#">Search.id</a> .	2015.2
<code>options.isPublic</code>	boolean <code>true</code>   <code>false</code>	Optional	<p>Set to <code>true</code> to make the search public. Otherwise, set to <code>false</code>. If you do not set this parameter, it defaults to <code>false</code>.</p> <p>This parameter sets the value for the <a href="#">Search.isPublic</a> property.</p>	2016.2

## Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	Required parameter is missing.
<code>SSS_INVALID_SRCH_FILTER_EXPR</code>	The <code>options.filters</code> parameter is not a valid search filter, filter array, or filter expression.
<code>SSS_INVALID_SRCH_COL</code>	The <code>options.columns</code> parameter is not a valid column, string, or column or string array.
<code>SSS_INVALID_SRCH_SETTING</code>	An unknown search parameter name is provided.
<code>SSS_INVALID_SRCH_SETTING_VALUE</code>	An unsupported value is set for the provided search parameter name.

## Syntax

**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 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)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a new search asynchronously and returns it as a <a href="#">search.Search</a> object.
	<p><b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">search.create(options)</a>. For more information on promises, see <a href="#">Promise Object</a>.</p>
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">search.create(options)</a>
<b>Supported Script Types</b>	All client-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/search Module</a>

Since	2015.2
-------	--------

## Syntax

**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
//Add additional code
...
search.create.promise({
    type: search.Type.SALES_ORDER
})
.then(function(result) {
    log.debug({
        details: "Completed: " + result
    });
    // do something after completion
})
.catch(function(reason) {
    log.debug({
        details: "Failed: " + reason
    })
    // do something on failure
});
...
//Add additional code
```

## search.createSetting(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a new search setting and returns it as a <a href="#">search.setting</a> object.  Search settings let you specify search parameters that are typically available only in the UI. The following settings are supported: <ul style="list-style-type: none"><li>■ <b>Consolidated Exchange Rate:</b> 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.</li><li>■ <b>Show Period End Transactions:</b> 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.</li></ul> After you create your settings, assign them as array values to <a href="#">Search.settings</a> .
<b>Returns</b>	<a href="#">search.setting</a>
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None

<b>Module</b>	N/search Module
<b>Since</b>	2018.2

## Parameters

 <b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
options.name	string	Required	<p>The name of the search parameter to set. This value sets the <a href="#">Setting.name</a> property.</p> <p>Use one of the following values for this parameter:</p> <ul style="list-style-type: none"> <li>▪ <b>consolidationtype</b>: This value corresponds to the Consolidated Exchange Rate setting.</li> <li>▪ <b>includeperiodendtransactions</b>: This value corresponds to the Show Period End Transactions setting.</li> </ul>	2018.2
options.value	string	Required	<p>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 <b>options.name</b> parameter. This value sets the <a href="#">Setting.value</a> property.</p> <p>If you specify <b>consolidationtype</b> as the search parameter name, use one of the following values for this parameter:</p> <ul style="list-style-type: none"> <li>▪ ACCTTYPE</li> <li>▪ AVERAGE</li> <li>▪ CURRENT</li> <li>▪ HISTORICAL</li> <li>▪ NONE</li> </ul> <p>The default value is ACCTTYPE, which represents the type of consolidation associated with the account.</p> <p>If you specify <b>includeperiodendtransactions</b> as the search parameter name, use one of the following values for this parameter:</p> <ul style="list-style-type: none"> <li>▪ F</li> <li>▪ FALSE</li> <li>▪ T</li> <li>▪ TRUE</li> </ul> <p>The default value is <b>false</b>.</p> <p>These values are not case sensitive.</p>	2018.2

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required parameter is missing.

Error Code	Thrown If
SSS_INVALID_SRCH_SETTING	An unknown search parameter name is provided.
SSS_INVALID_SRCH_SETTING_VALUE	An unsupported value is set for the provided search parameter name.

## Syntax

**⚠ 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 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)

**ⓘ Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Loads an existing saved search and returns it as a <a href="#">search.Search</a> . The saved search could have been created using the UI or created with <a href="#">search.create(options)</a> and <a href="#">Search.save()</a> .  The search.load(options) method also includes a promise version, <a href="#">search.load.promise(options)</a> . For more information about promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	<a href="#">search.Search</a>
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.id</code>	string	Required	Internal ID or script ID of a saved search. The script ID starts with <code>customsearch</code> . See <a href="#">Search.id</a> .	2015.2
<code>options.type</code>	string	Required if the saved search to load uses a standalone search type, optional otherwise	<p>The search type of the saved search to load. Use a value from the <a href="#">search.Type</a> enum for this parameter.</p> <p>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.</p> <p>The following is a list of standalone search types:</p> <ul style="list-style-type: none"> <li>■ DeletedRecord</li> <li>■ EndToEndTime</li> <li>■ ExpenseAmortPlanAndSchedule</li> <li>■ RevRecPlanAndSchedule</li> <li>■ GLinesAuditLog</li> <li>■ Crosschargeable</li> <li>■ FinRptAggregateFR</li> <li>■ BillingAccountBillCycle</li> <li>■ BillingAccountBillRequest</li> <li>■ BinItemBalance</li> <li>■ PaymentEvent</li> <li>■ Permission</li> <li>■ GatewayNotification</li> <li>■ TimeApproval</li> <li>■ RecentRecord</li> <li>■ Role</li> <li>■ SavedSearch</li> <li>■ ShoppingCart</li> <li>■ SubscriptionRenewalHistory</li> <li>■ SuiteScriptDetail</li> <li>■ SupplyChainSnapshotDetails</li> <li>■ SystemNote</li> <li>■ TaxDetail</li> <li>■ TimesheetApproval</li> <li>■ Uber</li> <li>■ ResAllocationTimeOffConflict</li> <li>■ ComSearchOneWaySyn</li> </ul>	2015.2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> <li>■ ComSearchGroupSyn</li> <li>■ Installment</li> <li>■ InventoryBalance</li> <li>■ InventoryNumberBin</li> <li>■ InventoryNumberItem</li> <li>■ InventoryStatusLocation</li> <li>■ InvNumberItemBalance</li> <li>■ ItemBinNumber</li> </ul>	

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	{1}: Missing a required argument: {2}	Required parameter is missing.
INVALID_SEARCH	That search or mass update does not exist.	Cannot find saved search with the saved search ID from <b>options.id</b> parameter.

## Syntax

 **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 mySearch = search.load({
    id: 'customsearch_my_so_search'
});
...
//Add additional code
```

## search.load.promise(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Loads an existing saved search asynchronously and returns it as a <a href="#">search.Search</a> object. The saved search could have been created using the UI or created with <a href="#">search.create(options)</a> and <a href="#">Search.save()</a> .
 <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">search.load(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .	
<b>Returns</b>	<a href="#">Promise Object</a>
<b>Synchronous Version</b>	<a href="#">search.load(options)</a>

<b>Supported Script Types</b>	All client-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

## Syntax

 **Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
//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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Deletes an existing saved search. The saved search could have been created using the UI or created with <a href="#">search.create(options)</a> and <a href="#">Search.save()</a> .  The search.delete(options) method also includes a promise version, search.delete.promise(options). For more information about promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	void
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	5 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

## Parameters

<b>Note:</b>	The options parameter is a JavaScript object.
--------------	---

Parameter	Type	Required / Optional	Description	Since
options.id	string	Required	Internal ID or script ID of a saved search. The script ID starts with <code>customsearch</code> . See <a href="#">Search.id</a> .	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	{1}: Missing a required argument: {2}	Required parameter is missing.
INVALID_SEARCH	That search or mass update does not exist.	Cannot find saved search with the saved search ID from <code>options.id</code> parameter.

## Syntax

<b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/search Module Script Samples</a> .
--

```
//Add additional code
...
search.delete({
  id: 'customsearch_my_so_search'
});
...
//Add additional code
```

## search.delete.promise(options)

<b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--------------	---

<b>Method Description</b>	Deletes an existing saved search asynchronously and returns it as a <code>search.Search</code> object. The saved search can be created using the UI or created with <code>search.create(options)</code> and <code>Search.save()</code> .  <b>Note:</b> The parameters and errors thrown for this method are the same as those for <code>search.delete(options)</code> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	<code>Promise Object</code>
<b>Synchronous Version</b>	<code>search.delete(options)</code>
<b>Supported Script Types</b>	All client-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	5 units

<b>Module</b>	N/search Module
<b>Since</b>	2015.2

## Syntax

**⚠ Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
//Add additional code
...
search.delete.promise({id: 'customsearch_txn_search_salesorder'})
    .then(function(){
        search.load({
            id: 'customsearch_txn_search_salesorder'
        });
    })
    .catch(function onRejected(reason) {
        log.debug({
            details: 'Invalid search: ' + reason.name
        });
    });
...
//Add additional code
```

## search.duplicates(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Performs a search for duplicate records based on the account's duplicate detection configuration.</p> <p>This method also includes a promise version, <a href="#">search.duplicates.promise(options)</a>. For more information about promises, see <a href="#">Promise Object</a>.</p> <p><b>⚠ Important:</b> This API works only for records that support duplicate record detection (for example, customer, lead, prospect, contact, partner, and vendor records).</p> <p>For more information about duplicate record detection, see the help topic <a href="#">Duplicate Record Detection</a>.</p>
<b>Returns</b>	<p><a href="#">search.Result[]</a> that contains the duplicate records</p> <p>Results are limited to 1000 rows.</p> <p>If there are no search results, this method returns <a href="#">null</a>.</p>
<b>Supported Script Types</b>	<p>All script types</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
<code>options.type</code>	enum	Required	<p>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).</p>	2015.2
<code>options.fields</code>	Object	Optional	<p>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 <code>'email' : 'sample@test.com'</code>.</p> <p>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 <code>options.id</code> parameter instead.</p>	2015.2
<code>options.id</code>	number	Optional	<p>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 <code>options.fields</code> parameter instead.</p>	2015.2

## Errors

Error Code	Message	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	{1}: Missing a required argument: {2}	Required parameter is missing.

## Syntax

 <b>Important:</b>	The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/search Module Script Samples</a> .
---	--

```
//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: {
        'email' : 'sample@test.com'
    }
});
...
//Add additional code

```

## search.duplicates.promise(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Performs a search for duplicate records asynchronously based on the Duplicate Detection configuration for the account. Returns an array of <a href="#">search.Result</a> objects. This method only applies to records that support duplicate record detection. These records include customer   lead   prospect   partner   vendor   contact.
	 <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">search.duplicates(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">search.duplicates(options)</a>
<b>Supported Script Types</b>	All client-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

### Syntax

 **Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```

//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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">Promise Object</a> .  For more information about global search, see the help topic <a href="#">Global Search</a> .
<b>Returns</b>	<code>search.Result[]</code> 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 <code>null</code> .
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
<code>options.keywords</code>	string	Required	Global search keywords string or expression.	2015.2

### Errors

Error Code	Message	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	{1}: Missing a required argument: {2}	Required 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/search Module Script Samples](#).

```
//Add additional code
...
var customerSearch = search.global({
    keywords: 'cu: simpson'
});
...
//Add additional code
```

## search.global.promise(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Performs a global search asynchronously against a single keyword or multiple keywords. Returns an array of <a href="#">search.Result</a> objects with four columns: <b>name</b> , <b>type</b> , <b>info1</b> , and <b>info2</b> .  <span style="background-color: #e0f2ff; border-radius: 10px; padding: 5px;"><b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">search.global(options)</a>. For more information on promises, see <a href="#">Promise Object</a>.</span>
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">search.global(options)</a>
<b>Supported Script Types</b>	All client-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

## Syntax



**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
//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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

Method Description	<p>Performs a search for one or more body fields on a record.</p> <p>You can use joined-field lookups with this method, with the following syntax:</p> <p><code>join_id.field_name</code></p> <p>The <code>search.lookupFields(options)</code> method also includes a promise version, <code>search.lookupFields.promise(options)</code>. For more information about promises, see <a href="#">Promise Object</a>.</p> <p>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.</p> <p>In the following example, a select field like <code>my_select</code> 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).</p> <p>For "internalid" in this particular code snippet, the sample returns <code>1234</code>. The internal id of a record is a single value, so a scalar is returned.</p> <pre> { internalid: 1234, firstname: 'Joe', my_select:   [{ value: 1, text: 'US Sub' }],   my_multiselect:   [ { value: 1, text: 'US Sub' },     { value: 2, text: 'EU Sub' } ] } </pre> <p>If you try to look up a field that does not exist on the specified record, an <code>SSS_INVALID_SRCH_COL</code> error is thrown.</p>
--------------------	---

<b>Returns</b>	Object   array <ul style="list-style-type: none"> <li>▪ Returns select fields as an object with value and text properties.</li> <li>▪ Returns multiselect fields as an array of object with value:text pairs.</li> </ul>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	1 unit
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.type	enum	Required	The search type for which you want to look up fields. Use the <a href="#">search.Type</a> enum for this argument.	2015.2
options.id	string	Required	Internal ID for the record, for example 777 or 87.	2015.2
options.columns	string   string[]	Required	Array of column/field names to look up, or a single column/field name. The <a href="#">columns</a> parameter can also be set to reference joined fields.	2015.2

## Errors

Error Code	Message	Thrown If
SSS_INVALID_SRCH_COL	An nlobjSearchColumn contains an invalid column, or is not in proper syntax: {1}.	The <a href="#">options.columns</a> parameter includes invalid columns for the specified record.
SSS_MISSING_REQD_ARGUMENT	{1}: Missing a required argument: {2}	Required 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/search Module Script Samples](#).

```
//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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
	 <b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">search.lookupFields(options)</a> . For more information on promises, see <a href="#">Promise Object</a> .
<b>Returns</b>	Promise Object
<b>Synchronous Version</b>	<a href="#">search.lookupFields(options)</a>
<b>Supported Script Types</b>	All client-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	1 unit
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

### Syntax

 **Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```

//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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a new search column as a <a href="#">search.Column</a> object.
	<p> <b>Important:</b> As you create search columns, consider the following:</p> <ul style="list-style-type: none"> <li>■ 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>.</li> <li>■ After you create a column, you cannot change the sort order of the column. If you use the same column in another search and specify a new sort order, the previous sort order is still used (the sort order that you specified using the <code>options.sort</code> parameter).</li> </ul>
<b>Returns</b>	<a href="#">search.Column</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.name</code>	string	Required	Name of the search column. See <a href="#">Column.name</a> .	2015.2
<code>options.join</code>	string	Optional	Join ID for the search column. See <a href="#">Column.join</a> .	2015.2
<code>options.summary</code>	enum	Optional	Summary type for the column. See <a href="#">search.Summary</a> and <a href="#">Column.summary</a> .	2015.2
<code>options.formula</code>	string	Optional	Formula for the search column. See <a href="#">Column.formula</a> .	2015.2
<code>options.function</code>	string	Optional	Special function for the search column. See <a href="#">Column.function</a> .	2015.2
<code>options.label</code>	string	Optional	Label for the search column. See <a href="#">Column.label</a> .	2015.2
<code>options.sort</code>	enum	Optional	The sort order of the column. Use the <a href="#">search.Sort</a> enum for this argument.	2015.2

Parameter	Type	Required / Optional	Description	Since
			Also see <a href="#">Column.sort</a> .	

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	{1}: Missing a required argument: {2}	Required parameter is missing.
SSS_INVALID_SRCH_COLUMN_SUM	A <a href="#">search.Column</a> object contains an invalid column summary type, or is not in proper syntax: {1}.	The <b>options.summary</b> parameter is not a valid search summary type. See <a href="#">search.Summary</a> .
INVALID_SRCH_FUNCTN		An unknown function is provided.

## Syntax

 **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 currencyColumn = search.createColumn({
    name: 'currency',
    sort: search.Sort.ASC
});
...
//Add additional code
```

## search.createFilter(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a new search filter as a <a href="#">search.Filter</a> object.
	 <b>Important:</b> 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> .
<b>Returns</b>	<a href="#">search.Filter</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.name	string	Required	Name or internal ID of the search field.	2015.2
options.join	string	Optional	Join ID for the search filter.	2015.2
options.operator	<a href="#">search.Operator</a>	Required	Operator used for the search filter. Use the <a href="#">search.Operator</a> enum.	2015.2
options.values	string   Date   number   boolean   string[]   Date[]   number[]	Optional	Values to be used as filter parameters.	2015.2
options.formula	string	Optional	Formula used by the search filter.	2015.2
options.summary	<a href="#">search.Summary</a>	Optional	Summary type for the search filter. See <a href="#">search.Summary</a> .	2015.2

## Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	{1}: Missing a required argument: {2}	Required parameter is missing.
SSS_INVALID_SRCH_FILTER_SUM	A <a href="#">search.Column</a> object contains an invalid column summary type, or is not in proper syntax: {1}.	options.summary parameter is not a valid search summary type. See <a href="#">search.Summary</a> .
SSS_INVALID_SRCH_OPERATOR	An <a href="#">search.Filter</a> object contains an invalid operator, or is not in proper syntax: {1}.	options.operator parameter is not a valid operator type. See <a href="#">search.Operator</a> .

## 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](#).

```
// 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	<p>Enumeration that holds the values for search operators to use with the <a href="#">search.Filter</a>.</p> <p>See the help topic <a href="#">SuiteScript 1.0 Search Operators</a> for more information about the field types supported for each operator type.</p>
<b>Supported Script Types</b>	<p>All script types</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

### Values

<ul style="list-style-type: none"> <li>■ AFTER</li> <li>■ ALLOF</li> <li>■ ANY</li> <li>■ ANYOF</li> <li>■ BEFORE</li> <li>■ BETWEEN</li> <li>■ CONTAINS</li> </ul>	<ul style="list-style-type: none"> <li>■ IS</li> <li>■ ISEMPTY</li> <li>■ ISNOT</li> <li>■ ISNOTEMPTY</li> <li>■ LESSTHAN</li> <li>■ LESSTHANOREQUALTO</li> <li>■ NONEOF</li> </ul>	<ul style="list-style-type: none"> <li>■ NOTGREATERTHANOREQUALTO</li> <li>■ NOTLESSTHAN</li> <li>■ NOTLESSTHANOREQUALTO</li> <li>■ NOTON</li> <li>■ NOTONORAFTER</li> <li>■ NOTONORBEFORE</li> <li>■ NOTWITHIN</li> </ul>
---	---	---

■ DOESNOTCONTAIN	■ NOTAFTER	■ ON
■ DOESNOTSTARTWITH	■ NOTALLOF	■ ONORAFTER
■ EQUALTO	■ NOTBEFORE	■ ONORBEFORE
■ GREATERTHAN	■ NOTBETWEEN	■ STARTSWITH
■ GREATERTHANOREQUALTO	■ NOTEQUALTO	■ WITHIN
■ HASKEYWORDS	■ NOTGREATERTHAN	

## Syntax

 **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 mySearchFilter = search.createFilter({
    name: 'entity',
    operator: search.Operator.ISEMPTY
});
...

//Add additional code
```

## search.Sort

<b>Enum Description</b>	Enumeration that holds the values for supported sorting directions used with <a href="#">search.createColumn(options)</a> .
	<p> <b>Note:</b> 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>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

## 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](#).

```
//Add additional code
```

```
...
var currencyColumn = search.createColumn({
  name: 'currency',
  sort: search.Sort.ASC
});
...
//Add additional code
```

## search.Summary

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the values for summary types used by the <a href="#">Column.summary</a> or <a href="#">Filter.summary</a> properties. For more information about each summary type, see the help topic <a href="#">SuiteScript 1.0 Search Summary Types</a> .
	 <b>Note:</b> 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.
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

### Values

- GROUP
- COUNT
- SUM
- AVG
- MIN
- MAX

### Syntax



**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 mySearchFilter = search.createFilter({
  name: 'entity',
  summary: search.Summary.GROUP
});
...
//Add additional code
```

## search.Type



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	<p>Enumeration that holds the string values for search types supported in the <a href="#">N/search Module</a>. This enum is used to pass the type argument to <a href="#">search.create(options)</a>.</p> <p><b>Note:</b> 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>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/search Module</a>
<b>Since</b>	2015.2

### Values



**Note:** A search type is not a synonym for a record type. The supported search types listed below do not necessarily correspond with the supported record types listed in the [N/record Module](#).

<ul style="list-style-type: none"> <li>■ ACCOUNT</li> <li>■ ACCOUNTING_BOOK</li> <li>■ ACCOUNTING_CONTEXT</li> <li>■ ACCOUNTING_PERIOD</li> <li>■ ACTIVITY</li> <li>■ ADV_INTER_COMPANY_JOURNAL_ENTRY</li> <li>■ AGGR_FIN_DAT</li> <li>■ AMORTIZATION_SCHEDULE</li> <li>■ AMORTIZATION_TEMPLATE</li> <li>■ ASSEMBLY_BUILD</li> <li>■ ASSEMBLY_ITEM</li> <li>■ ASSEMBLY_UNBUILD</li> <li>■ BILLING_ACCOUNT</li> <li>■ BILLING_ACCOUNT_BILL_CYCLE</li> <li>■ BILLING_ACCOUNT_BILL_REQUEST</li> <li>■ BILLING_CLASS</li> <li>■ BILLING_RATE_CARD</li> <li>■ BILLING_REVENUE_EVENT</li> <li>■ BILLING_SCHEDULE</li> <li>■ BIN</li> <li>■ BIN_ITEM_BALANCE</li> <li>■ BIN_TRANSFER</li> <li>■ BIN_WORKSHEET</li> <li>■ BLANKET_PURCHASE_ORDER</li> </ul>	<ul style="list-style-type: none"> <li>■ FIN_RPT_AGGREGATE_F_R</li> <li>■ FIXED_AMOUNT_PROJECT_REVENUE_RULE</li> <li>■ FOLDER</li> <li>■ FULFILLMENT_REQUEST</li> <li>■ GATEWAY_NOTIFICATION</li> <li>■ GENERIC_RESOURCE</li> <li>■ GIFT_CERTIFICATE</li> <li>■ GIFT_CERTIFICATE_ITEM</li> <li>■ GLOBAL_ACCOUNT_MAPPING</li> <li>■ GLOBAL_INVENTORY_RELATIONSHIP</li> <li>■ GL_LINES_AUDIT_LOG</li> <li>■ GL_NUMBERING_SEQUENCE</li> <li>■ GOAL</li> <li>■ INBOUND_SHIPMENT</li> <li>■ INSTALLMENT</li> <li>■ INTER_COMPANY_JOURNAL_ENTRY</li> <li>■ INTER_COMPANY_TRANSFER_ORDER</li> <li>■ INVENTORY_ADJUSTMENT</li> <li>■ INVENTORY_BALANCE</li> <li>■ INVENTORY_COST_REVALUATION</li> <li>■ INVENTORY_COUNT</li> <li>■ INVENTORY_DETAIL</li> <li>■ INVENTORY_ITEM</li> <li>■ INVENTORY_NUMBER</li> </ul>	<ul style="list-style-type: none"> <li>■ PROJECT_TEMPLATE</li> <li>■ PROMOTION_CODE</li> <li>■ PROSPECT</li> <li>■ PURCHASE_CONTRACT</li> <li>■ PURCHASE_ORDER</li> <li>■ PURCHASE_REQUSITION</li> <li>■ RECENT_RECORD</li> <li>■ RES_ALLOCATION_TIME_OFF_CONFLICT</li> <li>■ RESOURCE_ALLOCATION</li> <li>■ RESTLET</li> <li>■ RETURN_AUTHORIZATION</li> <li>■ REVENUE_ARRANGEMENT</li> <li>■ REVENUE_COMMITMENT</li> <li>■ REVENUE_COMMITMENT_REVERSAL</li> <li>■ REVENUE_PLAN</li> <li>■ REV_REC_PLAN_AND_SCHEDULE</li> <li>■ REV_REC_SCHEDULE</li> <li>■ REV_REC_TEMPLATE</li> <li>■ ROLE</li> <li>■ SALES_ORDER</li> <li>■ SALES_ROLE</li> <li>■ SALES_TAX_ITEM</li> <li>■ SAVED_SEARCH</li> <li>■ SCHEDULED_SCRIPT</li> </ul>
---	--	---

■ BOM	■ INVENTORY_NUMBER_BIN	■ SCHEDULED_SCRIPT_INSTANCE
■ BOM_REVISION	■ INVENTORY_NUMBER_ITEM	■ SCRIPT_DEPLOYMENT
■ BUDGET_EXCHANGE_RATE	■ INVENTORY_STATUS	■ SERIALIZED_ASSEMBLY_ITEM
■ BUNDLE_INSTALLATION_SCRIPT	■ INVENTORY_STATUS_CHANGE	■ SERIALIZED_INVENTORY_ITEM
■ CALENDAR_EVENT	■ INVENTORY_STATUS_LOCATION	■ SERVICE_ITEM
■ CAMPAIGN	■ INVENTORY_TRANSFER	■ SHIP_ITEM
■ CASH_REFUND	■ INVOICE	■ SHOPPING_CART
■ CASH_SALE	■ INVNT_NUMBER_ITEM_BALANCE	■ SOLUTION
■ CHARGE	■ ISSUE	■ STATISTICAL_JOURNAL_ENTRY
■ CHARGE_RULE	■ ITEM	■ STORE_PICKUP_FULFILLMENT
■ CHECK	■ ITEM_ACCOUNT_MAPPING	■ SUBSCRIPTION
■ CLASSIFICATION	■ ITEM_BIN_NUMBER	■ SUBSCRIPTION_CHANGE_ORDER
■ CLIENT_SCRIPT	■ ITEM_DEMAND_PLAN	■ SUBSCRIPTION_LINE
■ CMS_CONTENT	■ ITEM_FULFILLMENT	■ SUBSCRIPTION_PLAN
■ CMS_CONTENT_TYPE	■ ITEM_GROUP	■ SUBSCRIPTION_RENEWAL_HISTORY
■ CMS_PAGE	■ ITEM_LOCATION_CONFIGURATION	■ SUBSIDIARY
■ COM_SEARCH_BOOST	■ ITEM_RECEIPT	■ SUBTOTAL_ITEM
■ COM_SEARCH_BOOST_TYPE	■ ITEM_REVISION	■ SUITELET
■ COM_SEARCH_GROUP_SYN	■ ITEM_SUPPLY_PLAN	■ SUITE_SCRIPT_DETAIL
■ COM_SEARCH_ONE_WAY_SYN	■ JOB	■ SUPPLY_CHAIN_SNAPSHOT
■ COMMERCE_CATEGORY	■ JOB_STATUS	■ SUPPLY_CHAIN_SNAPSHOT_DETAILS
■ COMPETITOR	■ JOB_TYPE	■ SUPPORT_CASE
■ CONSOLIDATED_EXCHANGE_RATE	■ JOURNAL_ENTRY	■ SYSTEM_NOTE
■ CONTACT	■ KIT_ITEM	■ TASK
■ CONTACT_CATEGORY	■ LABOR_BASED_PROJECT_REVENUE_RULE	■ TAX_DETAIL
■ CONTACT_ROLE	■ LEAD	■ TAX_GROUP
■ COST_CATEGORY	■ LOCATION	■ TAX_PERIOD
■ COUPON_CODE	■ LOT_NUMBERED_ASSEMBLY_ITEM	■ TAX_TYPE
■ CREDIT_CARD_CHARGE	■ LOT_NUMBERED_INVENTORY_ITEM	■ TERM
■ CREDIT_CARD_REFUND	■ MANUFACTURING_COST_TEMPLATE	■ TIMESHEET_APPROVAL
■ CREDIT_MEMO	■ MANUFACTURING_OPERATION_TASK	■ TIME_APPROVAL
■ CROSSCHARGEABLE	■ MANUFACTURING_ROUTING	■ TIME_BILL
■ CURRENCY	■ MAP_REDUCE_SCRIPT	■ TIME_ENTRY
■ CUSTOMER	■ MARKUP_ITEM	■ TIME_OFF_CHANGE
■ CUSTOMER_CATEGORY	■ MASSUPDATE_SCRIPT	■ TIME_OFF_PLAN
■ CUSTOMER_DEPOSIT	■ MERCHANTISE_HIERARCHY_LEVEL	■ TIME_OFF_REQUEST
■ CUSTOMER_MESSAGE	■ MERCHANTISE_HIERARCHY_NODE	■ TIME_OFF_RULE
■ CUSTOMER_PAYMENT	■ MERCHANTISE_HIERARCHY_VERSION	■ TIME_OFF_TYPE
■ CUSTOMER_PAYMENT_AUTHORIZATION	■ MESSAGE	■ TIME_SHEET
■ CUSTOMER_REFUND	■ MFG_PLANNED_TIME	■ TOPIC
■ CUSTOMER_STATUS	■ NEXUS	■ TRANSACTION
■ CUSTOMER_SUBSIDIARY_RELATIONSHIP	■ NON_INVENTORY_ITEM	■ TRANSFER_ORDER
■ CUSTOM_RECORD	■ NOTE	■ UBER
■ CUSTOM_TRANSACTION	■ NOTE_TYPE	■ UNITS_TYPE
■ DELETED_RECORD	■ OPPORTUNITY	■ USAGE
■ DEPARTMENT	■ OTHER_CHARGE_ITEM	
	■ OTHER_NAME	

<ul style="list-style-type: none"> <li>■ DEPOSIT</li> <li>■ DEPOSIT_APPLICATION</li> <li>■ DESCRIPTION_ITEM</li> <li>■ DISCOUNT_ITEM</li> <li>■ DOWNLOAD_ITEM</li> <li>■ EMPLOYEE</li> <li>■ EMPLOYEE_CHANGE_REQUEST</li> <li>■ EMPLOYEE_CHANGE_TYPE</li> <li>■ END_TO_END_TIME</li> <li>■ ENTITY</li> <li>■ ENTITY_ACCOUNT_MAPPING</li> <li>■ ESTIMATE</li> <li>■ EXPENSE_AMORTIZATION_EVENT</li> <li>■ EXPENSE_AMORT_PLAN_AND_SCHEDULE</li> <li>■ EXPENSE_CATEGORY</li> <li>■ EXPENSE_PLAN</li> <li>■ EXPENSE_REPORT</li> <li>■ FAIR_VALUE_PRICE</li> </ul>	<ul style="list-style-type: none"> <li>■ OTHER_NAME_CATEGORY</li> <li>■ PARTNER</li> <li>■ PARTNER_CATEGORY</li> <li>■ PAYCHECK</li> <li>■ PAYCHECK_JOURNAL</li> <li>■ PAYMENT_EVENT</li> <li>■ PAYMENT_INSTRUMENT</li> <li>■ PAYMENT_ITEM</li> <li>■ PAYMENT_METHOD</li> <li>■ PAYROLL_ITEM</li> <li>■ PCT_COMPLETE_PROJECT_REVENUE_RULE</li> <li>■ PERFORMANCE REVIEW</li> <li>■ PERFORMANCE REVIEW_SCHEDULE</li> <li>■ PERIOD_END_JOURNAL</li> <li>■ PERMISSION</li> <li>■ PHONE_CALL</li> <li>■ PORTLET</li> <li>■ PRICE_BOOK</li> <li>■ PRICE_LEVEL</li> <li>■ PRICE_PLAN</li> <li>■ PRICING</li> <li>■ PRICING_GROUP</li> <li>■ PROJECT_EXPENSE_TYPE</li> <li>■ PROJECT_TASK</li> </ul>	<ul style="list-style-type: none"> <li>■ USEREVENT_SCRIPT</li> <li>■ VENDOR</li> <li>■ VENDOR_BILL</li> <li>■ VENDOR_CATEGORY</li> <li>■ VENDOR_CREDIT</li> <li>■ VENDOR_PAYMENT</li> <li>■ VENDOR_RETURN_AUTHORIZATION</li> <li>■ VENDOR_SUBSIDIARY_RELATIONSHIP</li> <li>■ WEBSITE</li> <li>■ WORKFLOW_ACTION_SCRIPT</li> <li>■ WORK_ORDER</li> <li>■ WORK_ORDER_CLOSE</li> <li>■ WORK_ORDER_COMPLETION</li> <li>■ WORK_ORDER_ISSUE</li> <li>■ WORKPLACE</li> </ul>
---	---	---

## Syntax



**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 mySearch = search.create({
    type: search.Type.CUSTOMER,
    filters: filters,
    columns: columns
});
...
//Add additional code
```

## N/sftp Module



**Note:** The content in this help topic pertains to SuiteScript 2.0.

The SFTP module provides a way to manage folders and upload or 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.

Use SSH keys to establish an SFTP connection. By using the keys, you can manage files and directories by using the SSH file transfer (SFTP) protocol. For more information, see the help topic [SSH Keys for SFTP](#). For more information about the N/keyControl Module, see [N/keyControl Module](#).



**Important:** All paths, directories, and filenames that contain wildcards such as ? and \* must have those characters escaped, unless these characters are specifically intended to work as wildcards.



**Note:** To use an external server to initiate a NetSuite file transfer that doesn't use SFTP, you can use RESTlets or SOAP 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 SOAP web services, applications can invoke CRUD operations on the file record to populate or change the contents of the File Cabinet. See the help topics [SuiteTalk SOAP Web Services Platform Guide](#) and [File](#) for more information.

- [N/sftp Module Members](#)
- [Connection Object Members](#)
- [N/sftp Module Script Samples](#)
- [Setting up an SFTP Transfer](#)
- [SFTP Authentication](#)
- [Supported Cipher Suites and Host Key Types](#)
- [Supported SuiteScript File Types](#)
- [N/keyControl Module](#)

## N/sftp Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">sftp.Connection</a>	Object	Server-side scripts	Represents a connection to the account on the remote FTP server.
Method	<a href="#">sftp.createConnection(options)</a>	<a href="#">sftp.Connection</a>	Server-side scripts	Establishes a connection to a remote FTP server.
Enum	<a href="#">sftp.MAX_CONNECT_TIMEOUT</a>	Enum	Server-side scripts	Holds the values for maximum connection timeout.
	<a href="#">sftp.MIN_CONNECT_TIMEOUT</a>	Enum	Server-side scripts	Holds the values for minimum connection timeout.
	<a href="#">sftp.MAX_PORT_NUMBER</a>	Enum	Server-side scripts	Holds the values for the maximum port number.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	sftp.MIN_PORT_NUMBER	Enum	Server-side scripts	Holds the values for the minimum port number.
	sftp.DEFAULT_PORT_NUMBER	Enum	Server-side scripts	Holds the values for the default port number.
	sftp.Sort	Enum	Server-side scripts	Holds the values to be used to sort listed directory.

## Connection Object Members

The following members are called on the `sftp.Connection` object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Connection.download(options)	file.File	Server-side scripts	Downloads a file from the remote FTP server.
	Connection.upload(options)	void	Server-side scripts	Uploads a file to the remote FTP server.
	Connection.makeDirectory(options)	string	Server-side scripts	Creates an empty directory.
	Connection.removeDirectory(options)	void	Server-side scripts	Removes an empty directory.
	Connection.removeFile(options)	void	Server-side scripts	Removes a file in a directory.
	Connection.move(options)	void	Server-side scripts	Moves a file or directory from one location to another.
	Connection.list(options)	array of objects	Server-side scripts	Lists the remote directory.
Enum	Connection.MAX_FILE_SIZE	number	Server-side scripts	Holds the values for the maximum file size.
	Connection.MAX_TRANSFER_TIMEOUT	number	Server-side scripts	Holds the values for the maximum transfer timeout.

## N/sftp Module Script Samples

**i Note:** These sample scripts use the `require` function so that you can copy it into the debugger and test them. 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](#).



**Important:** Before you run the 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 the sample, replace the placeholder values with valid values from your NetSuite account.

## Example 1



**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to upload and download 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 sample that shows creating a form field that generates a GUID.

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

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.Type.PLAINTEXT,
        contents: 'I am a test file.'
    });

    // upload the file to the remote server

    connection.upload({
        directory: 'relative/path/to/remote/dir',
        filename: 'newFileNameOnServer.js',
        file: myFileToUpload,
        replaceExisting: true
    });
});
```

```
// download the file from the remote server

var downloadedFile = connection.download({
    directory: 'relative/path/to/file',
    filename: 'downloadMe.js'
});

});
```

## Example 2

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how you can manage files and directories.

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

require(['N/sftp', 'N/file'], function(sftp, file){
    // Establish connection
    log.debug('Establishing SFTP connection...');
    var connection = sftp.createConnection({
        username: 'sftpuser',
        keyId: 'custkeysftp_nft_demo_key',
        url: 'myurl',
        port: 22,
        directory: 'inbound',
        hostKey: 'myhostkey'
    });
    log.debug('Connection established!');
    // -----
    // List the directory and log number of elements there
    var list = connection.list({path: 'yyy/test'});
    log.debug('Items in directory "test" at the beginning: ' + list.length);
    // -----
    // Generate the test file
    log.debug('Generating test file...');
    var myFileToUpload = file.create({
        name: 'asdf.txt',
        fileType: file.Type.PLAINTEXT,
        contents: 'I am a test file.'
    });
    log.debug('Test file generated, uploading to "test" directory...');
    // -----
    // Upload the test file
    connection.upload({
        directory: 'yyy/test',
        filename: 'af.txt',
        file: myFileToUpload,
        replaceExisting: true
    });
});
```

```

});  

log.debug('Upload complete!');  

// -----  

// List the directory to see there is one more file than before  

list = connection.list({path: 'yyy/test'});  

log.debug('Items in directory "test" after the upload: ' + list.length);  

// -----  

// Create new directory  

log.debug('Creating directory "test2"...');  

try {  

    connection.makeDirectory({path: 'yyy/test2'});  

    log.debug('Directory created.');  

} catch (e) {  

    log.debug('Directory not created.');  

    log.error(e.message);  

}  

list = connection.list({path: 'yyy/test2'});  

log.debug('Items in directory "test2": ' + list.length);  

// -----  

// Move the test file there  

log.debug('Moving the test file from "test" to "test2"...');  

connection.move({  

    from: 'yyy/test/af.txt',  

    to: 'yyy/test2/af.txt'  

})  

log.debug('File moved!');  

// -----  

// List the original directory again to see the file is moved out  

list = connection.list({path: 'yyy/test'});  

log.debug('Items in directory "test" after the upload: ' + list.length);  

// -----  

// List the new directory for the file  

list = connection.list({path: 'yyy/test2'});  

log.debug('Items in directory "test2" after the upload: ' + list.length);  

log.debug(JSON.stringify(list));  

// -----  

// Try to remove the directory  

log.debug('Removing directory "test2"...');  

try {  

    connection.removeDirectory({  

        path: 'yyy/test2'  

    });  

    log.debug('Directory removed!');  

} catch (e) {  

    log.debug('Directory not removed!');  

    log.error(e.message);  

}  

// -----  

// It's not empty so let's delete the file first  

log.debug('Removing test file from "test2" directory...');  

connection.removeFile({  

    path: 'yyy/test2/af.txt'  

});  

log.debug('Test file removed!');

```

```

list = connection.list({path: 'yyy/test2'});
log.debug('Items in directory "test2": ' + list.length);
// -----
// Remove it now again
log.debug('Removing directory "test2"...');
try {
    connection.removeDirectory({
        path: 'yyy/test2'
    });
    log.debug('Directory removed!');
} catch (e) {
    log.debug('Directory not removed!');
    log.error(e.message);
}
// -----
// Listing it again
log.debug('Trying to list directory "test2"...');
try {
    list = connection.list({path: 'yyy/test2'});
} catch (e) {
    log.error(e.message);
}
// -----
;
})

```

## Example 3

**i Note:** This script sample uses the **define** function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the **require** function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample shows how to use the N/sftp module enums to set conditional default settings. It creates a secure connection and attempts to upload and add a large file.

```

/**
 * @NApiVersion 2.x
 * @NScriptType UserEventScript
 * @NModuleScope SameAccount
 */

define(['N/file', 'N/sftp', 'N/error'],
function(file, sftp, error) {
    return {
        beforeLoad: function(){
            var portNumber = -1;
            var connectTimeout = -1;
            var transferTimeout = -1;
            //these variables can be taken as parameters of the script instead

            if (portNumber < sftp.MIN_PORT_NUMBER || portNumber > sftp.MAX_PORT_NUMBER)
                portNumber = sftp.DEFAULT_PORT_NUMBER;
        }
    }
})

```

```

if (connectTimeout < sftp.MIN_CONNECT_TIMEOUT) connectTimeout = sftp.MIN_CONNECT_TIMEOUT; else if
(connectTimeout > sftp.MAX_CONNECT_TIMEOUT)
    connectTimeout = sftp.MAX_CONNECT_TIMEOUT;

var connection = sftp.createConnection({
    username: 'sftpuser',
    keyId: 'custkey1',
    url: '192.168.0.100',
    port: portNumber,
    directory: ' inbound',
    timeout: connectTimeout,
    hostKey: "AAAAB3NzaC1yc2EAAAQABAAQDMifKH2vTxdiyep8nem7+1S3x7dTQR/A67KdsR/5C2WUcDipBzYhH
bnG6Am12Nd2t1M01LnaBZA6/8P4Y9x/sGTxtsdE/MzeGDUBn6HB1QvgIrhX62wgoKGQ+P2lEA01+Vz8y3/MB1NmD7Fc62cJ9Mu88YA6jwJOIPZ
eHYNVyIm90rY6VyzYyvSJh0x7SXvGnijJQF4G8C4c8u/UVpF/sE16xKZtly2Rx0aDL2FsDRtpyPmM602/R6ISbsmgab3MzzAEIu+zLDMdIB
Jn3cDhNt1F7RaZ6Tu0u18KCKk8GPxbnxDUG4sCNOnXPYkDXSMUbM/oCrjYGtqdZUMmeTf3"
});

// can also be a big file (created for example by async search)
var myFileToUpload = file.create({
    name: 'originalname.txt',
    fileType: file.Type.PLAINTEXT,
    contents: 'I am a test file.'
});

if (myFileToUpload.size > connection.MAX_FILE_SIZE)
    throw error.create({name:"FILE_IS_TOO_BIG", message:"The file you are trying to upload is too
big"});

if (transferTimeout > connection.MAX_TRANSFER_TIMEOUT)
    transferTimeout = connection.MAX_TRANSFER_TIMEOUT;
else if (transferTimeout < connection.MIN_TRANSFER_TIMEOUT)
    transferTimeout = connection.MIN_TRANSFER_TIMEOUT;

connection.upload({
    directory: 'files',
    filename: 'test.txt',
    file: myFileToUpload,
    replaceExisting: true,
    timeout: transferTimeout
});

}
};

});
```

## 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/key
    - 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.

## 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

## 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



**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();
    response.writePage(form);
}
...
...
```

### Reading the Credential Token in a Suitelet

Note that the following code snippet is not a fully functional sample.

```
...
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. For more information, see the help topic [SSH Keys for SFTP](#).

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
AATpn1P9jB+cQx9Jq9UeZjA1245X7SBDCrKh+Sok56VzSw==
```

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:

Communication protocol	<p>SFTP (SSH + FTP) is supported.</p> <p>Only CTR (and not CBC) ciphers are allowed. Your SFTP server can use the following encryption algorithms:</p> <ul style="list-style-type: none"> <li>■ AES 128-CTR</li> <li>■ AES 192-CTR</li> <li>■ AES 256-CTR</li> <li>■ RSA</li> <li>■ DSA</li> <li>■ ECDSA</li> </ul> <p>Files are not additionally encrypted during transfer. The entire transmission is encrypted by the SSH protocol.</p>
Authentication mechanism	<p>Username</p> <p>Password</p> <p>Password/SSH key with or without passphrase</p>
SSH host key	With each connection request, you must supply the host key. Any host key changes need to be managed manually.
GUID	<p>The password GUID should be a value generated by a credential field from a Suitelet using <a href="#">Form.addCredentialField(options)</a>.</p> <p>The password GUID field's originating credential field must include the SFTP domain on the <code>restrictToDomains</code> parameter.</p> <p>The password GUID field's originating credential field must include the script utilizing the password GUID on the <code>restrictToScriptIds</code> parameter.</p>

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.

...

```
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](#)).

You can also create and remove directories. For more information, see [N/sftp Module Members](#).

## Syntax

```
require(['N/sftp', 'N/file'],
  function (sftp, file)
{
  var connection = sftp.createConnection({
    /*
     * The Username supplied by the administrator of the external SFTP server.
     */
    username: 'myuser',
    /*
     * Refers to the Password supplied by the administrator of the external SFTP server.
     *
     * The Password Token/GUID obtained by reading the form POST parameter associated
     * with user submission of a form containing a Credential Field.

    Value would typically be read from a custom field.
    */
    passwordGUID: "B34672495064525E5D65032D63B52301",
    /*
     * The URL supplied by the administrator of the external SFTP server.
     */
    url: 'host.somewhere.com',
    /*
     * The SFTP Port number supplied by the administrator of the external SFTP server (defaults to 22).
     */
    port: 22,
    /*
     * The transfer directory supplied by the administrator of the external SFTP server (optional).
     */
    directory: 'transferfiles',
    /*
     * RSA Host Key obtained via ssh-keyscan tool.

     $ ssh-keyscan -t rsa -p 22 host.somewhere.com
     AATpn1P9jB+cQx9Jq9UeZjA1245X7SBDCRiKh+Sok56VzSw==
     */
    hostKey: "AATpn1P9jB+cQx9Jq9UeZjA1245X7SBDCRiKh+Sok56VzSw=="
  });
}
```

```

/*
Creating a simple file.
*/
var myFileToUpload = file.create({
    name: 'originalname.js',
    fileType: file.Type.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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Represents a connection to the account on the remote FTP server.
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Module</b>	N/sftp Module
<b>Methods and Properties</b>	Connection Object Members
<b>Since</b>	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/sftp Module Script Samples](#).

```
//Add additional code ...
// establish connection to the FTP server
var objConnection = sftp.createConnection({
    username: 'username',
    keyId: 'custkey1',
    url: 'host.somewhere.com',
    directory: 'username/wheres/my/file'
    hostKey: myHostKey
});
...
//Add additional code
```

## Connection.download(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Downloads a file from the remote FTP server.
<b>Returns</b>	<a href="#">file.File</a> Object
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	100
<b>Module</b>	N/sftp Module
<b>Since</b>	2016.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.filename	string	Required	The name of the file to download.	2016.2
options.directory	string	Optional	The relative path to the directory that contains the file to download.  By default, the path is set to the current directory.	2016.2

Parameter	Type	Required / Optional	Description	Since
			 <b>Important:</b> This input must take the form of a relative path.	
options.timeout	number	Optional	The number of seconds to allow for the file to download.  By default, this value is set to 300 seconds.	2016.2

## Errors

Error Code	Thrown If
FTP_MAXIMUM_FILE_SIZE_EXCEEDED	The file size is greater than the maximum file size allowed by NetSuite.
FTP_NO_SUCH_FILE_OR_DIRECTORY	The file or directory does not exist.
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_PERMISSION_DENIED	Access to the file or directory on the remote FTP server was denied.
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.

## Syntax

 **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 Samples](#).

```
//Add additional code
...
var downloadedFile = objConnection.download({
    directory: 'relative/path/to/file',
    filename: 'downloadMe.js'
});
...
//Add additional code
```

## Connection.upload(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Uploads a file to the remote FTP server. The maximum file size that can be uploaded to is 100 MB.
---------------------------	---

<b>Returns</b>	void
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	100
<b>Module</b>	<a href="#">N/sftp Module</a>
<b>Since</b>	2016.2

## Parameters

 <b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
<code>options.file</code>	<code>file.File</code>	Required	The file to upload.	2016.2
<code>options.filename</code>	string	Optional	The name to give the uploaded file on server.  By default, the filename is the same specified by <code>options.file</code> .   <b>Note:</b> Illegal characters are automatically escaped.	2016.2
<code>options.directory</code>	string	Optional	The relative path to the directory where the file should be upload to.  By default, the path is set to the current directory.   <b>Important:</b> This input must take the form of a relative path.	2016.2
<code>options.timeout</code>	number	Optional	The number of seconds to allow for the file to upload.  By default, this value is set to 300 seconds.   <b>Important:</b> This parameter does not specify the overall timeout limit. The value is only applied when no data is received within the specified period.	2016.2
<code>options.replaceExisting</code>	boolean <code>true</code>   <code>false</code>	Optional	Indicates whether the file being uploaded should overwrite any file with the name <code>options.filename</code> that already exists in <code>options.directory</code> .  If <code>false</code> , the <code>FTP_FILE_ALREADY_EXISTS</code> exception is thrown when a file with the same name already exists in the <code>options.directory</code> .	2016.2

Parameter	Type	Required / Optional	Description	Since
			By default, this value is false.	

## Errors

Error Code	Thrown If
FTP_NO_SUCH_FILE_OR_DIRECTORY	The file or directory does not exist.
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 Samples](#).

```
//Add additional code ...
objConnection.upload({
    directory: 'relative/path/to/remote/dir',
    filename: 'newFileNameOnServer',
    file: myFileToUpload,
    replaceExisting: true
});
...
//Add additional code
```

## Connection.makeDirectory(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates an empty directory.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server-side scripts

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10
<b>Module</b>	N/sftp Module
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.path	string	Required	The relative path of a directory to be created.	2019.2

## Errors

Error Code	Thrown If
FTP_PERMISSION_DENIED	Access to the file or directory on the remote FTP server was denied.
FTP_DIRECTORY_NOT_FOUND	Creating a directory in a non-existent directory.

## Syntax



**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 Samples](#).

```
// Add additional code ...
objConnection.makeDirectory({
    directory: 'relative/path/to/remote/dir',
});
...
//Add additional code
```

## Connection.removeDirectory(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Removes an empty directory.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10

<b>Module</b>	N/sftp Module
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.path	String	Required	The relative path of a directory to be deleted.	2019.2

## Errors

Error Code	Thrown If
FTP_PERMISSION_DENIED	Access to the file or directory on the remote FTP server was denied.
FTP_DIRECTORY_NOT_FOUND	Deleting a directory that does not exist.
FTP_DIRECTORY_NOT_EMPTY	Deleting a directory that is not empty.

## Syntax

 **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 Samples](#).

```
// Add additional code ...
objConnection.removeDirectory({
    directory: 'relative/path/to/remote/dir',
});
...
//Add additional code
```

## Connection.removeFile(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Removes a file.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10
<b>Module</b>	N/sftp Module

Since	2019.2
-------	--------

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.path	String	Required	The relative path of the directory location where this file should be removed.	2019.2

## Errors

Error Code	Thrown If
FTP_PERMISSION_DENIED	Access to the file or directory on the remote FTP server was denied.
FTP_NO_SUCH_FILE_OR_DIRECTORY	The file or directory does not exist.

## Syntax

 **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 Samples](#).

```
//Add additional code ...
objConnection.removeFile({
    directory: 'relative/path',
});
...
//Add additional code
```

## Connection.move(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Moves a file or directory from one location to another.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10
<b>Module</b>	<a href="#">N/sftp Module</a>
<b>Since</b>	2019.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.from	String	Required	The relative path of the file to be moved from.	2019.2
options.to	String	Required	The relative path of the file to be moved to.	2019.2

## Errors

Error Code	Thrown If
FTP_INVALID_MOVE	Source is not readable or the target is not writable. Source or target does not exist.

## Syntax

 **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 Samples](#).

```
// Add additional code ...
objConnection.move({
  from: 'relative/path/to/remote/dir',
  to: 'relative/path/to/remote/dir/new',
});
...
// Add additional code
```

## Connection.list(options)

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Lists the remote directory.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10
<b>Module</b>	<a href="#">N/sftp Module</a>
<b>Since</b>	2019.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.path	String	Required	The relative path to directory of file that will be downloaded.	2019.2
options.sort	String	Required	The sort options. Values from the sort enum are accepted.	2019.2

## Errors

Error Code	Thrown If
FTP_INVALID_DIRECTORY	The directory does not exist on the remote FTP server.
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 Samples](#).

```
// Add additional code ...
var connection = sftp.createConnection({
    username: 'sftpuser',
    keyId:"custkey2",
    url: '172.25.184.111',
    port: 22,
    directory: '',
    hostKey: "hostkey"
});
// Add additional code
```

**i Note:** Wildcards are accepted. The ? symbol can represent any character. The \* symbol can represent any number of characters.

## Connection.MAX\_FILE\_SIZE

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the values for the maximum file size.
	<b>i Note:</b> 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.
<b>Type</b>	enum

<b>Module</b>	N/sftp Module
<b>Sibling Module Members</b>	N/sftp Module Members
<b>Since</b>	2019.2

## Values

- 100000000

## Syntax

See [N/sftp Module Script Samples](#).

## Connection.MAX\_TRANSFER\_TIMEOUT

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the values for the maximum transfer timeout.
	<p><b>Note:</b> 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>
<b>Type</b>	enum
<b>Module</b>	N/sftp Module
<b>Sibling Module Members</b>	N/sftp Module Members
<b>Since</b>	2019.2

## Values

- 300

## Syntax

See [N/sftp Module Script Samples](#).

## sftp.createConnection(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Establishes a connection to a remote FTP server.
---------------------------	--

	<p>To generate the <code>passwordguid</code>, you can create a suitelet that uses <code>Form.addCredentialField(options)</code>.</p> <p>Use the <a href="#">N/https Module</a> to fetch the GUID value returned from the Suitelet's credential field.</p> <p>For more information, see <a href="#">Setting up an SFTP Transfer</a> and <a href="#">Supported Cipher Suites and Host Key Types</a>.</p>
<b>Returns</b>	<code>sftp.Connection</code> , representing that connection.
<b>Supported Script Types</b>	All server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/sftp Module</a>
<b>Since</b>	2016.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
<code>options.url</code>	string	Required	The host of the remote account.	2016.2
<code>options.passwordGuid</code>	string	Required	The password GUID for the remote account.  This is only required if key ID is not provided.	2016.2
<code>options.hostKey</code>	string	Required	The host key for the trusted fingerprint on the server.  The host key is required even if keyId is supplied instead of passwordGuid.	2016.2
<code>options.username</code>	string	Required	The username of the remote account.  By default, the login is anonymous.	2016.2
<code>options.port</code>	number	Optional	The port used to connect to the remote account.  By default, port 22 is used.	2016.2
<code>options.directory</code>	string	Optional	The remote directory of the connection.   <b>Note:</b> The directory property is required if you use a remote server cannot resolve relative paths.	2016.2
<code>options.timeout</code>	number	Optional	The number of seconds to allow for an established connection.  By default, this value is set to 20 seconds.	2016.2
<code>options.hostKeyType</code>	string	Optional	The type of host key specified by <code>options.hostKey</code> .	2016.2

Parameter	Type	Required / Optional	Description	Since
			This value can be set to one of the following options: ■ dsa ■ ecdsa ■ rsa	
options.keyId	string	Optional	The ID of the key to be used for authentication.	2016.2

## Errors

Error Code	Thrown If
FTP_UNKNOWN_HOST	The host could not be found.
FTP_CONNECT_TIMEOUT_EXCEEDED	A connection could not be established within options.timeout seconds.
FTP_CANNOT_ESTABLISH_CONNECTION	The password/username was invalid or permission to access the directory was denied.
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_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_ATTEMPTS	There are too many incorrect authentication attempts.
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 Samples](#).

```
//Add additional code
...
// establish connection to the ftp server

var objConnection = sftp.createConnection({
    username: 'username',
    keyId: 'custkey1',
    url: 'host.somewhere.com',
    directory: 'username/wheres/my/file'
    hostKey: myHostKey // references var myHostKey
});
...
//Add additional code
```

## sftp.MAX\_CONNECT\_TIMEOUT

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the values for the maximum connection timeout.
<b>Type</b>	enum
<b>Module</b>	<a href="#">N/sftp Module</a>
<b>Sibling Module Members</b>	<a href="#">N/sftp Module Members</a>
<b>Since</b>	2016.2

## Values

- 20

## Syntax

See [N/sftp Module Script Samples](#).

## sftp.MIN\_CONNECT\_TIMEOUT

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the values for the minimum connection timeout.
-------------------------	--

<b>Type</b>	enum
<b>Module</b>	N/sftp Module
<b>Sibling Module Members</b>	N/sftp Module Members
<b>Since</b>	2016.2

## Values

- 1

## Syntax

See [N/sftp Module Script Samples](#).

sftp.MAX\_PORT\_NUMBER

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Enum Description</b>	Holds the values for the maximum port number.
<b>Type</b>	enum
<b>Module</b>	N/sftp Module
<b>Sibling Module Members</b>	N/sftp Module Members
<b>Since</b>	2019.2

## Values

- 65535

## Syntax

See [N/sftp Module Script Samples](#).

sftp.MIN\_PORT\_NUMBER

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Enum Description</b>	Holds the values for the minimum port number.
<b>Type</b>	enum
<b>Module</b>	N/sftp Module

<b>Sibling Module Members</b>	N/sftp Module Members
<b>Since</b>	2019.2

## Values

- 0

## Syntax

See [N/sftp Module Script Samples](#).

## sftp.DEFAULT\_PORT\_NUMBER

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Enum Description</b>	Holds the values for default port number.
<b>Type</b>	enum
<b>Module</b>	<a href="#">N/sftp Module</a>
<b>Sibling Module Members</b>	<a href="#">N/sftp Module Members</a>
<b>Since</b>	2019.2

## Values

- 22

## Syntax

See [N/sftp Module Script Samples](#).

## sftp.Sort

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Enum Description</b>	Holds the values to be used to sort the listed directory.
	 <b>Note:</b> 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.
<b>Type</b>	enum

<b>Module</b>	N/sftp Module
<b>Sibling Module Members</b>	N/sftp Module Members
<b>Since</b>	2019.2

## Values

- DATE
- DATE\_DESC
- SIZE
- SIZE\_DESC
- NAME
- NAME\_DESC

## Syntax

**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 Samples](#).

```
//Add additional code ...
require(['N/sftp'],function(sftp){
var connection = sftp.createConnection({
    username: 'sftpuser',
    keyId: 'custkey1',
    url: '192.168.0.100',
    port: 22,
    directory: 'inbound',
    hostKey: "AAAAB3NzaC1yc2EAAAQABAAQDMifKH2vTxdiy8nem7+ls3x7dTQR/A67KdsR/5C2WUcdipBzYhHbnG6Am1
2Nd2t1M01LnA6/8P4Y9x/sGtxsdE/MzeGDUBn6HBlQvgIrhX62wgoKGQ+P21EA01+Vz8y3/MB1NmD7Fc62cJ9Mu88YA6jwJOIPZeHYNVyI
m90rY6VyzYvsJhH0x7SyvGniijQF4G8C4c8u/UVpF/sE16xKztly2Rx0aDL2FsDRtpyPmM602/R6ISbsmgab3MzzAEIu+zLDMdIBJn3cDhN
t1F7Rar6Tu0u18KCkk8GPxbnxDuG4sCNOnXPYkDXSMUbM/ocRjYGtqdZUMmeTf3"
});
connection.list({
    path:"?path*",
    sort: sftp.Sort.SIZE});
})
//Add additional code
```

## N/sso Module

**Note:** The content in this help topic pertains to SuiteScript 2.0.

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

Member Type	Name	Return Type	Supported Script Types	Description
Method	<a href="#">sso.generateSuiteSignOnToken(options)</a>	string	Portlet scripts, user event scripts, and Suitelets	Generates a new SuiteSignOn token for a user

## N/sso Module Script Sample

 **Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

 **Important:** The value used in this sample for the `suiteSignOnRecordId` field is a placeholder. Before using this sample, replace the `suiteSignOnRecordId` field value with a valid value from your NetSuite account. If you run a script with an invalid value, an error may occur. 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](#).

The following sample shows how to generate a new OAuth token for a user. This sample requires the SuiteSignOn feature.

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

require(['N/sso'],function(sso) {
    function generateSSOToken() {
        var suiteSignOnRecordId = 1;
        var url = sso.generateSuiteSignOnToken(suiteSignOnRecordId);
    }
    generateSSOToken();
});
```

 **Note:** This script sample uses the `define` function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the `require` function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample shows how to use `generateSuiteSignOnToken(options)` in a portlet script.



**Important:** The value used in this sample for the suiteSignOnRecordId field is a placeholder. Before using this sample, replace the suiteSignOnRecordId field value with a valid value from your NetSuite account. If you run a script with an invalid value, an error may occur. 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](#).

```
/*
 * @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
    };
});
```



**Note:** This script sample uses the **define** function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the **require** function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample shows how to use **generateSuiteSignOnToken(options)** in a Suitelet script.



**Important:** The value used in this sample for the suiteSignOnRecordId field is a placeholder. Before using this sample, replace the suiteSignOnRecordId field value with a valid value from your NetSuite account. If you run a script with an invalid value, an error may occur. 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](#).

```
/*
 * @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
    };
});
```

```
});
```

**Note:** This script sample uses the **define** function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the **require** function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample shows how to use **generateSuiteSignOnToken(options)** in a user event script.

**Important:** The value used in this sample for the suiteSignOnRecordId field is a placeholder. Before using this sample, replace the suiteSignOnRecordId field value with a valid value from your NetSuite account. If you run a script with an invalid value, an error may occur. 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](#).

```
/**  
 * @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  
    };  
});
```

## sso.generateSuiteSignOnToken(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to generate a new SuiteSignOn token for a user.
	<p><b>Note:</b> To use this method, Outbound Single Sign-on and SOAP web services must be enabled in your account. To enable these features, go to Setup &gt; Company &gt; Enable Features. On the SuiteCloud tab, in the Manage Authentication section, select the SuiteSignOn check box. In the SuiteTalk section, select the SOAP Web Services check box. Click Save.</p>
<b>Returns</b>	URL, OAuth token, and any integration variables as a string
<b>Supported Script Types</b>	Portlet scripts, user event scripts, and Suitelets For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	20 units

<b>Module</b>	N/sso Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options. suiteSignOnId	string	required	<p>The scriptId specified on the SuiteSignOn record. To see a list of IDs for SuiteSignOn records, go to the SuiteSignOn list page (Setup &gt; Integration &gt; SuiteSignOn ).</p> <p> <b>Note:</b> 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.</p>	2015.2

## Errors

Error Code	Message	Thrown If
INVALID_SSO	Invalid SuiteSignOn reference: {1}. That SuiteSignOn object does not exist or has been marked as inactive.	The <b>suiteSignOnId</b> input parameter is invalid or does not exist.
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 <b>suiteSignOnId</b> 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](#).

```
//Add additional code
...
var suiteSignOnRecordId = 1;
var url = sso.generateSuiteSignOnToken('customsso1');
...
//Add additional code
```

# N/task Module



**Note:** The content in this help topic pertains to SuiteScript 2.0.

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 Object Members](#)
- [RecordActionTaskStatus Object Members](#)
- [N/task Module Script Samples](#)

## N/task Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">task.ScheduledScriptTask</a>	Object	Server scripts	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.
	<a href="#">task.ScheduledScriptTaskStatus</a>	Object	Server scripts	Encapsulates the status of a scheduled script placed into the NetSuite scheduling queue.
	<a href="#">task.MapReduceScriptTask</a>	Object	Server scripts	Encapsulates a map/reduce script deployment.
	<a href="#">task.MapReduceScriptTaskStatus</a>	Object	Server scripts	Encapsulates the status of a map/reduce script deployment that has been submitted for processing.
	<a href="#">task.CsvImportTask</a>	Object	Server scripts	Encapsulates the properties of a CSV import task. Use the methods and properties for this object to submit a CSV

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
				import task into the task queue and asynchronously import record data into NetSuite.
	task.CsvImportTaskStatus	Object	Server scripts	Encapsulates the status of a CSV import task placed into the NetSuite scheduling queue.
	task.EntityDeduplicationTask	Object	Server scripts	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.
	task.EntityDeduplicationTaskStatus	Object	Server scripts	Encapsulates the status of a merge duplicate record task placed into the NetSuite task queue.
	task.SearchTask	Object	Server scripts	Encapsulates the properties required to initiate an asynchronous search.
	task.SearchTaskStatus	Object	Server scripts	Encapsulates the status of an asynchronous search initiation task that is placed into the NetSuite task queue.
	task.WorkflowTriggerTask	Object	Server scripts	Encapsulates all the properties required to asynchronously initiate a workflow. Use WorkflowTriggerTask to create a task that initiates an instance of a specific workflow.
	task.WorkflowTriggerTaskStatus	Object	Server scripts	Encapsulates the status of an asynchronous workflow initiation task placed into the NetSuite task queue.
	task.RecordActionTask	Object	Server scripts	Encapsulates the properties of a record action task. Use this object to place a record action task into the NetSuite scheduling queue.
	task.RecordActionTaskStatus	Object	Server scripts	Encapsulates the status of a record action task in the NetSuite scheduling queue.
Method	task.create(options)	task.ScheduledScriptTask   task.MapReduceScriptTask   task.CsvImportTask   task.EntityDeduplicationTask   task.SearchTask   task.WorkflowTriggerTask	Server scripts	Creates an object for a specific task type and returns the task object.
	task.checkStatus(options)	task.RecordActionTask   task.ScheduledScriptTaskStatus   task.MapReduceScriptTaskStatus   task.CsvImportTaskStatus   task.EntityDeduplicationTaskStatus   task.SearchTaskStatus   task.WorkflowTriggerTaskStatus   task.RecordActionTaskStatus	Server scripts	Returns a task status object associated with a specific task ID.
Enum	task.TaskType	enum	Server scripts	Enumeration that holds the string values for the types of task objects, supported by the N/task

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
				Module, that you can create with <a href="#">task.create(options)</a> .
	task.TaskStatus	enum	Server scripts	Enumeration that holds the string values for the possible status of tasks created and submitted with the <a href="#">N/task Module</a> .
	task.MasterSelectionMode	enum	Server scripts	Enumeration that holds the string values for supported master selection modes when merging duplicate records with <a href="#">task.EntityDeduplicationTask</a> .
	task.DedupeMode	enum	Server scripts	Enumeration that holds the string values for available deduplication modes when merging duplicate records with <a href="#">task.EntityDeduplicationTask</a> .
	task.DedupeEntityType	enum	Server scripts	Enumeration that holds the string values for entity types for which you can merge duplicate records with <a href="#">task.EntityDeduplicationTask</a> .
	task.MapReduceStage	enum	Server scripts	Enumeration that holds the string values for the stages of a map/reduce script deployment, which is encapsulated by the <a href="#">task.MapReduceScriptTask</a> object.
	task.ActionCondition	enum	Server scripts	Enumeration that holds the string values for the possible record action conditions.

## ScheduledScriptTask Object Members

The following members are called on [task.ScheduledScriptTask](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">ScheduledScriptTask.submit()</a>	string	Server scripts	Directs NetSuite to place a scheduled script deployment into the NetSuite scheduling queue and returns a unique ID for the task.
Property	<a href="#">ScheduledScriptTask.scriptId</a>	number   string	Server scripts	Internal ID (as a number), or script ID (as a string) for the script record associated with a <a href="#">task.ScheduledScriptTask</a> object.
	<a href="#">ScheduledScriptTask.deploymentId</a>	string	Server scripts	Script ID (as a string), for the script deployment record associated with a <a href="#">task.ScheduledScriptTask</a> object.
	<a href="#">ScheduledScriptTask.params</a>	Object	Server scripts	Object with key/value pairs that override the static script parameter field values on the script deployment.

## ScheduledScriptTaskStatus Object Members

The following members are called on [task.ScheduledScriptTaskStatus](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	ScheduledScriptTaskStatus.scriptId	read-only number	Server scripts	Internal ID for a script record associated with a specific <code>task.ScheduledScriptTask</code> object.
	ScheduledScriptTaskStatus.deploymentId	read-only script ID	Server scripts	Script ID for a script deployment record associated with a specific <code>task.ScheduledScriptTask</code> object.
	ScheduledScriptTaskStatus.task.TaskStatus	<code>task.TaskStatus</code>	Server scripts	Status for a scheduled script task. Returns a <code>task.TaskStatus</code> enum value.

## MapReduceScriptTask Object Members

The following members are called on `task.MapReduceScriptTask`.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<code>MapReduceScriptTask.submit()</code>	string	Server scripts	Submits a map/reduce script deployment for processing.
Property	MapReduceScriptTask.scriptId	number   string	Server scripts	Internal ID (as a number), or script ID (as a string), for the map/reduce script record.
	MapReduceScriptTask.deploymentId	string	Server scripts	Script ID (as a string), for the script deployment record for a map/reduce script.
	MapReduceScriptTask.params	Object	Server scripts	Object that represents key/value pairs that override static script parameter field values on the script deployment record.

## MapReduceScriptTaskStatus Object Members

The following members are called on the `task.MapReduceScriptTaskStatus` object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<code>MapReduceScriptTaskStatus.getPercentageCompleted()</code>	number	Server scripts	Returns the current percentage complete for the current stage of a <code>task.MapReduceScriptTask</code> .
	<code>MapReduceScriptTaskStatus.getPendingMapCount()</code>	number	Server scripts	Returns the total number of records or rows not yet processed by the map stage of a <code>task.MapReduceScriptTask</code> .
	<code>MapReduceScriptTaskStatus.getTotalMapCount()</code>	number	Server scripts	Returns the total number of records or rows passed as input to the map stage of a <code>task.MapReduceScriptTask</code> .
	<code>MapReduceScriptTaskStatus.getPendingMapSize()</code>	number	Server scripts	Returns the total number of bytes not yet processed by the map stage, as a

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
				component of total size, of a <a href="#">task.MapReduceScriptTask</a> .
	<a href="#">MapReduceScriptTask Status.getPendingReduceCount()</a>	number	Server scripts	Returns the total number of records or rows not yet processed by the reduce stage of a <a href="#">task.MapReduceScriptTask</a> .
	<a href="#">MapReduceScriptTask Status.getTotalReduceCount()</a>	number	Server scripts	Returns the total number of record or row inputs to the reduce stage of a <a href="#">task.MapReduceScriptTask</a> .
	<a href="#">MapReduceScriptTask Status.getPendingReduceSize()</a>	number	Server scripts	Returns the total number of bytes not yet processed by the reduce stage, as a component of total size, of a <a href="#">task.MapReduceScriptTask</a> .
	<a href="#">MapReduceScriptTask Status.getPendingOutputCount()</a>	number	Server scripts	Returns the total number of records or rows not yet processed by a <a href="#">task.MapReduceScriptTask</a> .
	<a href="#">MapReduceScriptTask Status.getPendingOutputSize()</a>	number	Server scripts	Returns the total size in bytes of all key/value pairs written as output, as a component of total size, by a <a href="#">task.MapReduceScriptTask</a> .
	<a href="#">MapReduceScriptTask Status.getTotalOutputCount()</a>	number	Server scripts	Returns the total number of records or rows passed as inputs to the output phase of a <a href="#">task.MapReduceScriptTask</a> .
	<a href="#">MapReduceScriptTask Status.getCurrentTotalSize()</a>	number	Server scripts	Returns the total size in bytes of all stored work in progress by a <a href="#">task.MapReduceScriptTask</a> .
Property	<a href="#">MapReduceScriptTask Status.scriptId</a>	read-only number   string	Server scripts	Internal ID for a map/reduce script record associated with a specific <a href="#">task.MapReduceScriptTask</a> .
	<a href="#">MapReduceScriptTask Status.deploymentId</a>	read-only string	Server scripts	Script ID for a script deployment record associated with a specific <a href="#">task.MapReduceScriptTask</a> .
	<a href="#">MapReduceScriptTask Status.status</a>	<a href="#">task.TaskStatus</a>	Server scripts	Status for a map/reduce script task. Returns a <a href="#">task.TaskStatus</a> enum value.
	<a href="#">MapReduceScriptTask Status.stage</a>	<a href="#">task.MapReduceStage</a>	Server scripts	The current stage of a map/reduce script deployment that is being processed. See <a href="#">task.MapReduceStage</a> for supported values.

## CsvImportTask Object Members

The following members are called on [task.CsvImportTask](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">CsvImportTask.submit()</a>	string	Server scripts	Directs NetSuite to place a CSV import task into the NetSuite task queue and returns a unique ID for the task.
Property	<a href="#">CsvImportTask.importFile</a>	file.File   string	Server scripts	CSV file to import. Use a <a href="#">file.File</a> object or a string that represents the CSV text to be imported.
	<a href="#">CsvImportTask.mappingId</a>	number   string	Server scripts	Script ID or internal ID of the saved import map that you created when you ran the Import Assistant.
	<a href="#">CsvImportTask.queueId</a>	number	Server scripts	Overrides the <b>Queue Number</b> property under <b>Advanced Options</b> on the <b>Import Options</b> page of the Import Assistant.
	<a href="#">CsvImportTask.name</a>	string	Server scripts	Name for the CSV import task.
	<a href="#">CsvImportTask.linkedFiles</a>	Object	Server scripts	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.

## CsvImportTaskStatus Object Members

The following members are called on [task.CsvImportTaskStatus](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	<a href="#">CsvImportTaskStatus.status</a>	<a href="#">task.TaskStatus</a>	Server scripts	Status for a CSV import task. Returns a <a href="#">task.TaskStatus</a> enum value.

## EntityDeduplicationTask Object Members

The following members are called on [task.EntityDeduplicationTask](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">EntityDeduplicationTask.submit()</a>	string	Server scripts	Directs NetSuite to place the merge duplicate records task into the NetSuite task queue and returns a unique ID for the task.
Property	<a href="#">EntityDeduplicationTask.entityType</a>	<a href="#">task.DedupeEntityType</a>	Server scripts	Sets the type of entity on which you want to merge duplicate records.
	<a href="#">EntityDeduplicationTask.masterRecordId</a>	number	Server scripts	When you merge duplicate records, you can delete all duplicates for a record or merge information from the duplicate records into the master record.
	<a href="#">EntityDeduplicationTask.masterSelectionMode</a>	<a href="#">task.MasterSelectionMode</a>	Server scripts	When you merge duplicate records, you can delete all

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
				duplicates for a record or merge information from the duplicate records into the master record.
	EntityDeduplicationTask.dedupeMode	task.DedupeMode	Server scripts	Sets the mode in which to merge or delete duplicate records.
	EntityDeduplicationTask.recordIds	number[]	Server scripts	Number array of record internal IDs to perform the merge or delete operation on.

## EntityDeduplicationTaskStatus Object Members

The following members are called on [task.EntityDeduplicationTaskStatus](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	<a href="#">EntityDeduplicationTaskStatus.status</a>	task.TaskStatus	Server scripts	Status for a merge duplicate record task.

## SearchTask Object Members

The following members are called on [task.SearchTask](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">SearchTask.addInboundDependency()</a>	void	Server scripts	Adds a scheduled script task or map/reduce script task 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 <a href="#">SuiteCloud Processors</a> .
	<a href="#">SearchTask.submit()</a>	string	Server scripts	Places the asynchronous search initiation task into the SuiteScript task queue, and returns a unique ID for the task.
Property	<a href="#">SearchTask.fileId</a>	number	Server scripts	ID of the CSV file to export search results into.
	<a href="#">SearchTask.filePath</a>	string	Server scripts	Path of the CSV file to export search results into.
	<a href="#">SearchTask.inboundDependencies</a>	Object	Server scripts	Object that contains key/value pairs to describe the dependent scripts added to the search task.
	<a href="#">SearchTask.savedSearchId</a>	number	Server scripts	ID of the saved search to be executed during the task.

## SearchTaskStatus Object Members

The following members are called on [task.SearchTaskStatus](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	SearchTaskStatus.fileId	number	Server scripts	ID of the CSV file into which search results are exported.
	SearchTaskStatus.savedSearchId	number	Server scripts	ID of the saved search executed during the task.
	SearchTaskStatus.status	task.TaskStatus	Server scripts	Status of an asynchronous search task placed in the NetSuite task queue.
	SearchTaskStatus.taskId	number	Server scripts	ID of the asynchronous task.

## WorkflowTriggerTask Object Members

The following members are called on [task.WorkflowTriggerTask](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">WorkflowTriggerTask.submit()</a>	string	Server scripts	Directs NetSuite to place the asynchronous workflow initiation task into the NetSuite scheduling queue and returns a unique ID for the task.
Property	<a href="#">WorkflowTriggerTask.recordType</a>	string	Server scripts	Record type of the workflow base record. For example, customer, salesorder, or lead.
	<a href="#">WorkflowTriggerTask.recordId</a>	number	Server scripts	Internal ID of the workflow definition base record. For example, 55 or 124.
	<a href="#">WorkflowTriggerTask.workflowId</a>	number   string	Server scripts	Internal ID (as a number), or script ID (as a string), for the workflow definition.
	<a href="#">WorkflowTriggerTask.params</a>	Object	Server scripts	Object that contains key/value pairs to set default values on fields specific to the workflow.

## WorkflowTriggerTaskStatus Object Members

The following members are called on the [task.WorkflowTriggerTaskStatus](#) object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	<a href="#">WorkflowTriggerTaskStatus.status</a>	task.TaskStatus	Server scripts	Status for a asynchronous workflow placed in the NetSuite task queue.

## RecordActionTask Object Members

The following members are called on [task.RecordActionTaskStatus](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	RecordActionTask.submit()	string	Server scripts	Submits a record action task for processing and returns its task ID.
	RecordActionTask.toString()	string	Server scripts	Returns the object type name.
	RecordActionTask.toJSON()	Object	Server scripts	Returns an object in JSON.
Property	RecordActionTask.recordType	string	Server scripts	The record type on which the action is to be performed. For a list of record types, see <a href="#">record.Type</a> .
	RecordActionTask.paramCallback()	Object	Server scripts	Function that takes record ID and returns the parameter object for the specified record ID.
	RecordActionTask.action	string	Server scripts	The ID of the action to be invoked.
	RecordActionTask.params	Array of objects	Server scripts	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'}
	RecordActionTask.condition	Object	Server scripts	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.

## RecordActionTaskStatus Object Members

The following members are called on [task.RecordActionTaskStatus](#).

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	RecordActionTaskStatus.toString()	string	Server scripts	Returns the object type name.
	RecordActionTaskStatus.toJSON()	Object	Server scripts	Returns an object in JSON.
Property	RecordActionTaskStatus.status	string	Server scripts	Represents the record action task status. Returns a value from the <a href="#">task.TaskStatus</a> enum.
	RecordActionTaskStatus.results	Object	Server scripts	The results of successfully executed record action tasks. The value of the property is the task instance ID and the corresponding action result.
	RecordActionTaskStatus.errors	Object	Server scripts	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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
				object with two properties: code and message.
	RecordActionTask Status.complete	number	Server scripts	The number of record action tasks with a completed status.
	RecordActionTask Status.succeeded	number	Server scripts	The number of record action tasks with a succeeded status.
	RecordActionTask Status.failed	number	Server scripts	The number of record action tasks with a failed status.
	RecordActionTask Status.pending	number	Server scripts	The number of record action tasks with a pending status.

## N/task Module Script Samples

The following script samples demonstrate how to use the features of the N/task module.

### Sample 1: Create and submit a map/reduce script

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample submits a map/reduce script task for processing. Before you use this sample, you must create a map/reduce script file, upload the file to your NetSuite account, and create a script record and script deployment record for it. For help working with map/reduce scripts, see the help topic [SuiteScript 2.0 Map/Reduce Script Type](#). You must also edit the sample and replace all hard-coded IDs with values that are valid in your NetSuite account.

```
/** 
 * @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: Create and submit an asynchronous search task

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following 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 sample retrieves the task status using the task ID. Some of the values in this 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.

```

/**
 * @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 my fileId = 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: Create and submit a task with dependent scripts



**Note:** This sample script uses the **require** function so that you can copy it into the SuiteScript Debugger and test it. You must use the **define** function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following 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](#).

This 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 samples below. Before using this sample, create these parameters in the script record. For more information, see the help topic [Creating Script Parameters](#).

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

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 in 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) {
    // Load the search results file and send an email with the file attached and
    // the number of rows in the file

    function execute(context) {
        // Read a CSV file and return the number of rows minus the header row
        function numberOfRows(csv fileId) {
            var invoiceFile = file.load({
                id: csv fileId
            });
            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 numberOfRows;
    }

    // Send an email to the user who ran the script, and attach the
    // CSV file with the search results
    function sendEmailWithAttachment(csv fileId) {
        var noOfRows = numberOfRows(csv fileId);
        var userId = runtime.getCurrentUser().id;
        var fileObj = file.load({
            id: csv fileId
        });

        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 res fileId = runtime.getCurrentScript().getParameter({
    name: 'custscript_ss_as_srch_res'
});

```

```

    if (!resFileDialog) {
        log.error('Could not obtain file content from the specified ID.');
        return;
    }

    log.debug({
        title: 'search - numberOfRows',
        details: numberOfRows(resFileDialog)
    });
    sendEmailWithAttachment(resFileDialog);
}

return {
    execute: execute
};
});

```

To read the contents of the search results file in a dependent map/reduce script, consider the following script sample:

```

/**
 * @NApiVersion 2.x
 * @NScriptType MapReduceScript
 * @NModuleScope SameAccount
 */
define(['N/runtime', 'N/file', 'N/log', 'N/email'], function(runtime, file, log, email) {
    // Load the search results file, count the number of letters in the file, and
    // store this count in another file

    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 resFileDialog = runtime.getCurrentScript().getParameter({
            name: completionScriptParameterName
        });

        if (!resFileDialog) {
            log.error({
                details: 'resFileDialog is not valid. Please check the script parameter stored in the completionScriptParameterName variable in getInputData().'
            });
        }

        return {
            type: 'file',
            id: resFileDialog
        };
    }

    function map(context) {
        var email = context.value.split(',')[1];

```

```

        if ("Email" !== email) {
            var splitEmail = email.split('@');
            context.write(splitEmail[splitEmail.length-1], 1);
        }
    }

    function reduce(context) {
        context.write(context.key, context.values.length);
    }

    function summarize(summary) {
        var type = summary.toString();
        log.audit({title: type + ' Usage Consumed ', details: summary.usage});
        log.audit({title: type + ' Concurrency Number ', details: summary.concurrency});
        log.audit({title: type + ' Number of Yields ', details: summary.yields});

        var contents = '';
        summary.output.iterator().each(function(key, value) {
            contents += (key + ' ' + value + '\n');
            return true;
        });
    }

    // Create the output file
    //
    // Update the name parameter to use the file name of the output file
    var fileObj = file.create({
        name: 'domainCount.txt',
        fileType: file.Type.PLAINTEXT,
        contents: contents
    });

    // Specify the folder location of the output file, and save the file
    //
    // Update the fileObj.folder property with the ID of the folder in
    // the file cabinet that contains the output file
    fileObj.folder = -15;
    fileObj.save();
}

return {
    getInputData: getInputData,
    map: map,
    reduce: reduce,
    summarize: summarize
};
});
}
);

```

## Sample 4: Submit a task and check its status

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to submit a record action task and then check its status. For details about record action tasks, see [task.RecordActionTask](#) and [task.RecordActionTaskStatus](#).

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

require(['N/task'], function(task) {
    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);
});
```

## task.ScheduledScriptTask

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	<p>Encapsulates all the properties of scheduled script task in SuiteScript. Use this object to place a scheduled script deployment into the NetSuite scheduling queue.</p> <p>To use the <code>ScheduledScriptTask</code> Object:</p> <ol style="list-style-type: none"> <li>1. In the NetSuite UI, create the script record and script deployment record.</li> <li>2. Use <code>task.create(options)</code> to create the <code>ScheduledScriptTask</code> object.</li> <li>3. Use the <code>ScheduledScriptTask</code> object properties to set the script and deployment properties.</li> <li>4. Use <code>ScheduledScriptTask.submit()</code> to deploy the scheduled script to the NetSuite scheduling queue.</li> <li>5. Use the properties for the <code>task.ScheduledScriptTaskStatus</code> object to get the status of the scheduled script.</li> </ol> <p>For a complete list of this object's methods and properties, see <a href="#">ScheduledScriptTask Object Members</a>.</p>
---------------------------	--

	For more information about scheduled scripts in NetSuite, see the help topic <a href="#">SuiteScript 2.0 Scheduled Script Type</a> .
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//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();
...
//Add additional code
```

## ScheduledScriptTask.submit()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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: <ul style="list-style-type: none"><li>■ The scheduled script must have a status of <b>Not Scheduled</b> on the Script Deployment page. If the script status is set to <b>Testing</b> on the Script Deployment page, this method will not place the script into the scheduling queue.</li><li>■ If the deployment status on the Script Deployment page is set to <b>Scheduled</b>, the script will be placed into the queue according to the time(s) specified on the Script Deployment page.</li><li>■ Only roles with the SuiteScript Scheduling permission or Administrators can run scheduled scripts executed by API. If a user event script calls <b>ScheduledScriptTask.submit()</b>, the user event script must be initiated by a role with permission.</li><li>■ 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 <b>submit()</b> method to resubmit itself, then at runtime, no task ID is returned when the scheduled script is submitted.</li></ul>
<b>Returns</b>	The task ID as a string, except as noted above.
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Governance</b>	20 units
<b>Module</b>	N/task Module
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
FAILED_TO_SUBMIT_JOB_REQUEST_1	Failed to submit job request: {reason}	Task cannot be submitted.

## Syntax

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

```
//Add additional code
...
var scheduledScriptTaskId = scriptTask.submit();
...
//Add additional code
```

## ScheduledScriptTask.scriptId

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Internal ID (as a number), or script ID (as a string), for the script record associated with a <a href="#">task.ScheduledScriptTask</a> object.
<b>Type</b>	number   string
<b>Governance</b>	20 units
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/task Module
<b>Since</b>	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 task Module Script Sample.

```
//Add additional code
...
var scheduledscriptId = 34;
...
//Add additional code
```

## ScheduledScriptTask.deploymentId



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Script ID (as a string), for the script deployment record associated with a <a href="#">task.ScheduledScriptTask</a> Object.
<b>Type</b>	string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//Add additional code .
...
scheduledTask.deploymentId = custdeploy1;
...
//Add additional code
```

## ScheduledScriptTask.params



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Object with key/value pairs that override static script parameter field values on the script deployment. Use these parameters for the <a href="#">task.ScheduledScriptTask</a> object to programmatically pass values to the script deployment.  For more information about script parameters, see the help topic <a href="#">Creating Script Parameters Overview</a> .
<b>Type</b>	object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//Add additional code
```

```
...
scriptTask.params = {searchId: 'custsearch_456'};
...
//Add additional code
```

## task.ScheduledScriptTaskStatus



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the properties and status of a scheduled script placed into the NetSuite scheduling queue.  Use <a href="#">task.checkStatus(options)</a> with the unique ID for the scheduled script task to get the <b>ScheduledScriptTaskStatus</b> Object.  For a complete list of this object's properties, see <a href="#">ScheduledScriptTaskStatus Object Members</a> .
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//Add additional code
...
var res = task.checkStatus(scriptTaskId);
log.debug('Initial status: ' + res.status);
...
//Add additional code
```

## ScheduledScriptTaskStatus.scriptId



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Internal ID for a script record associated with a specific <a href="#">task.ScheduledScriptTask</a> Object.  Use this ID to get more details about the script record for the scheduled task.
<b>Type</b>	read-only number
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>

<b>Since</b>	2015.2
--------------	--------

## Errors

Error Code	Message	Thrown If
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 task Module Script Sample.

```
//Add additional code
...
log.audit('Initial status: ' + status.scriptId);
...
//Add additional code
```

## ScheduledScriptTaskStatus.deploymentId

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Script ID for a script deployment record associated with a specific <a href="#">task.ScheduledScriptTask</a> Object.  Use this ID to get more details about the script deployment record for the scheduled task.
<b>Type</b>	string
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
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 task Module Script Sample.

```
//Add additional code
...
log.audit({
  details:'Deployment ID: ' + status.scriptId
```

```
});  
...  
//Add additional code
```

## ScheduledScriptTaskStatus.status

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Status for a scheduled script task. Returns a <a href="#">task.TaskStatus</a> enum value.
<b>Type</b>	<a href="#">task.TaskStatus</a>
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2015.2

### Errors

Error Code	Message	Thrown If
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 [task Module Script Sample](#).

```
//Add additional code  
...  
log.audit({  
    details: 'Status: ' + summary.status  
});  
...  
//Add additional code
```

## task.MapReduceScriptTask

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	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 <b>MapReduceScriptTask</b> object: <ul style="list-style-type: none"><li>■ In the NetSuite UI, create the script record and script deployment records.</li><li>■ Use <a href="#">task.create(options)</a> to create the <b>MapReduceScriptTask</b> object.</li><li>■ Use the <b>MapReduceScriptTask</b> object properties to set the script and deployment properties.</li></ul>
---------------------------	--

	<ul style="list-style-type: none"> <li>■ Use <a href="#">MapReduceScriptTask.submit()</a> to submit the deployment for processing.</li> <li>■ Use the properties for the <a href="#">task.MapReduceScriptTaskStatus</a> object to get the status of the map/reduce script.</li> </ul> <p>For a complete list of this object's methods and properties, see <a href="#">MapReduceScriptTask Object Members</a>.</p> <p>For general information about map/reduce scripts, see the help topic <a href="#">Map/Reduce Key Concepts</a>.</p>
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//Add additional code
...
var mrTask = task.create({taskType: task.TaskType.MAP_REDUCE});
mrTask.scriptId = mapReducescriptId;
mrTask.deploymentId = custdeploy1;
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()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Submits a map/reduce script deployment for processing.</p> <p>For more information, see <a href="#">task.MapReduceScriptTask</a>.</p> <p>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.</p> <p>Executing a map/reduce script from a server script requires either the Administrator role or a role with the SuiteScript Scheduling permission.</p> <p>For general information about the execution of map/reduce scripts, see the help topic <a href="#">SuiteScript 2.0 Map/Reduce Script Submission</a>.</p>
<b>Returns</b>	string

<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	20 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
FAILED_TO_SUBMIT_JOB_REQUEST_1	Failed to submit job request: {reason}	Task cannot be submitted.

## Syntax



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

```
//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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Internal ID (as a number), or script ID (as a string), for the map/reduce script record.
<b>Type</b>	number   string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//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

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Script ID (as a string), for the script deployment record for a map/reduce script.
<b>Type</b>	string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//Add additional code
...
mrTask.deploymentId = custdeploy1;
...
//Add additional code
```

**Note:** For general information about map/reduce scripts, see the help topic [SuiteScript 2.0 Map/Reduce Script Type](#).

## MapReduceScriptTask.params

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Object that represents key/value pairs that override static script parameter field values on the script deployment record.  Use these parameters on a <a href="#">task.MapReduceScriptTask</a> object to programmatically pass values to the script deployment. For more information about script parameters, see the help topic <a href="#">Creating Script Parameters Overview</a> .
<b>Type</b>	object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>

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 task Module Script Sample.

```
//Add additional code
...
mrTask.params = {doSomething: true};
...
//Add additional code
```

**Note:** For general information about map/reduce scripts, see the help topic [SuiteScript 2.0 Map/Reduce Script Type](#).

## task.MapReduceScriptTaskStatus

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the status of a map/reduce script deployment that was submitted for processing.  Use <a href="#">task.checkStatus(options)</a> with the unique ID for the map/reduce script task to get the <b>MapReduceScriptTaskStatus</b> object.  For a complete list of this object's methods and properties, see <a href="#">MapReduceScriptTaskStatus Object Members</a> .  For general information about the execution of map/reduce scripts, see the help topic <a href="#">SuiteScript 2.0 Map/Reduce Script Submission</a> .
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
if (summary.stage === task.MapReduceStage.SUMMARIZE)
    log.audit('Almost done...');

...
//Add additional code
```



**Note:** For general information about map/reduce scripts, see the help topic [SuiteScript 2.0 Map/Reduce Script Type](#).

## MapReduceScriptTaskStatus.getPercentageCompleted()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the current percentage complete for the current stage of a <a href="#">task.MapReduceScriptTask</a> .  Use the <a href="#">MapReduceScriptTaskStatus.stage</a> property to get the current stage.  For general information about map/reduce stages, see the help topics <a href="#">Map/Reduce Key Concepts</a> and <a href="#">SuiteScript 2.0 Map/Reduce Script Stages</a> .
<b>Returns</b>	number
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//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()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the total number of records or rows not yet processed by the map stage of a <a href="#">task.MapReduceScriptTask</a> .
---------------------------	--

	Use the <a href="#">MapReduceScriptTaskStatus.stage</a> property to get the current stage.  For general information about map/reduce stages, see the help topics <a href="#">Map/Reduce Key Concepts</a> and <a href="#">SuiteScript 2.0 Map/Reduce Script Stages</a> .
<b>Returns</b>	number
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//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()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the total number of records or rows passed as input to the map stage of a <a href="#">task.MapReduceScriptTask</a> .  Use the <a href="#">MapReduceScriptTaskStatus.stage</a> property to get the current stage.  For general information about map/reduce stages, see the help topics <a href="#">Map/Reduce Key Concepts</a> and <a href="#">SuiteScript 2.0 Map/Reduce Script Stages</a> .
<b>Returns</b>	number
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//Add additional code
...
var summary = taskStatus.getTotalMapCount();
log.audit('Total 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.getPendingMapSize()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the total number of bytes not yet processed by the map stage, as a component of total size, of a <a href="#">task.MapReduceScriptTask</a> .  Use the <a href="#">MapReduceScriptTaskStatus.stage</a> property to get the current stage.  For general information about map/reduce stages, see the help topics <a href="#">Map/Reduce Key Concepts</a> and <a href="#">SuiteScript 2.0 Map/Reduce Script Stages</a> .
<b>Returns</b>	number
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	25 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//Add additional code
...
var summary = taskStatus.getPendingMapSize();
log.audit('Pending Map Size: ' + summary);
...
//Add additional code
```



**Note:** For general information about map/reduce scripts, see the help topic [SuiteScript 2.0 Map/Reduce Script Type](#).

## MapReduceScriptTaskStatus.getPendingReduceCount()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the total number of records or rows not yet processed by the reduce stage of a <a href="#">task.MapReduceScriptTask</a> .  Use the <a href="#">MapReduceScriptTaskStatus.stage</a> property to get the current stage.  For general information about the reduce stage and other map/reduce stages, see the help topics <a href="#">Map/Reduce Key Concepts</a> and <a href="#">SuiteScript 2.0 Map/Reduce Script Stages</a> .
<b>Returns</b>	number
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//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()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the total number of record or row inputs to the reduce stage of a <a href="#">task.MapReduceScriptTask</a> .  Use the <a href="#">MapReduceScriptTaskStatus.stage</a> property to get the current stage.  For general information about map/reduce stages, see the help topics <a href="#">Map/Reduce Key Concepts</a> and <a href="#">SuiteScript 2.0 Map/Reduce Script Stages</a> .
---------------------------	---

<b>Returns</b>	number
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//Add additional code
...
var summary = taskStatus.getTotalReduceCount();
log.audit({
    details: '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.getPendingReduceSize()

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the total number of bytes not yet processed by the reduce stage, as a component of total size, of a <a href="#">task.MapReduceScriptTask</a> . Use the <a href="#">MapReduceScriptTaskStatus.stage</a> property to get the current stage.
<b>Returns</b>	number
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	25 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

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

```

var summary = taskStatus.getPendingReduceSize();
log.audit({
  details: 'Pending Reduce Size: ' + summary
});
...
//Add additional code

```

**Note:** For general information about map/reduce scripts, see the help topic [SuiteScript 2.0 Map/Reduce Script Type](#).

## MapReduceScriptTaskStatus.getPendingOutputCount()

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the total number of records or rows not yet processed by a <a href="#">task.MapReduceScriptTask</a> .
<b>Returns</b>	number
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```

//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
var total = summary.getPendingOutputCount()
log.audit({
  title: 'Count',
  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.getPendingOutputSize()

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the total size in bytes of all key/value pairs written as output, as a component of total size, by a <a href="#">task.MapReduceScriptTask</a> .
---------------------------	---

<b>Returns</b>	number
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	25 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
var total = summary.getPendingOutputSize()
log.audit({
    title: 'Size',
    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.getTotalOutputCount()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the total number of key/value pairs passed as inputs to the <b>SUMMARIZE</b> phase of a <a href="#">task.MapReduceScriptTask</a> . Use the <a href="#">MapReduceScriptTaskStatus.stage</a> property to get the current stage.
<b>Returns</b>	number
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//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()

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the total size in bytes of all stored work in progress by a <a href="#">task.MapReduceScriptTask</a> .
<b>Returns</b>	number
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	25 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Internal ID for a map/reduce script record associated with a specific <code>task.MapReduceScriptTask</code> .
<b>Type</b>	read-only number
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2015.2

### Errors

Error Code	Message	Thrown If
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 [task Module Script Sample](#).

```
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
log.audit({
    title: 'Script ID',
    details: summary.scriptId
});

...
//Add additional code
```



**Note:** For general information about map/reduce scripts, see the help topic [SuiteScript 2.0 Map/Reduce Script Type](#).

## MapReduceScriptTaskStatus.deploymentId



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Script ID for a script deployment record associated with a specific <code>task.MapReduceScriptTask</code> .
-----------------------------	---

Type	read-only string
Supported Script Types	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Module	N/task Module
Since	2015.2

## Errors

Error Code	Message	Thrown If
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 [task Module Script Sample](#).

```
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
log.audit({
    title: 'Deployment ID',
    details: summary.deploymentId
});
...
//Add additional code
```

**ℹ Note:** For general information about map/reduce scripts, see the help topic [SuiteScript 2.0 Map/Reduce Script Type](#).

## MapReduceScriptTaskStatus.status

**ℹ Note:** The content in this help topic pertains to SuiteScript 2.0.

Property Description	Status of a map/reduce script deployment that was submitted for processing. Returns a <a href="#">task.TaskStatus</a> enum value.  For general details about the execution of map/reduce scripts, see the help topic <a href="#">SuiteScript 2.0 Map/Reduce Script Submission</a> .
Type	<a href="#">task.TaskStatus</a>
Supported Script Types	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Module	N/task Module
Since	2015.2

## Errors

Error Code	Message	Thrown If
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 task Module Script Sample.

```
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
log.audit({
  title: 'Status',
  details: summary.status
});

...
//Add additional code
```

**i Note:** For general information about map/reduce scripts, see the help topic [SuiteScript 2.0 Map/Reduce Script Type](#).

## MapReduceScriptTaskStatus.stage

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Current stage of processing for a map/reduce script deployment instance. See <a href="#">task.MapReduceStage</a> for supported values.  For general information about map/reduce stages, see the help topic <a href="#">Map/Reduce Key Concepts</a> . For information about the execution of map/reduce scripts, see the help topic <a href="#">SuiteScript 2.0 Map/Reduce Script Submission</a> .
<b>Type</b>	<a href="#">task.MapReduceStage</a>
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	<p>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.</p> <p>Use the <b>CsvImportTask</b> Object to perform the following types of tasks:</p> <ul style="list-style-type: none"> <li>■ Automate standard record data import for SuiteApp installations, demo environments, and testing environments.</li> <li>■ Import data on a schedule using a scheduled script.</li> <li>■ Build integrated CSV imports with RESTlets.</li> </ul> <p>Use the following process to import CSV data with CsvImportTask:</p> <ul style="list-style-type: none"> <li>■ 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: <ul style="list-style-type: none"> <li>□ Script ID for import map.</li> <li>□ Any required linked files.</li> </ul> For more information, see the help topic <a href="#">Importing CSV Files with the Import Assistant</a>.</li> <li>■ Use <a href="#">task.create(options)</a> to create the CsvImportTask object.</li> <li>■ Use the CsvImportTask object properties to set the script and deployment properties.</li> <li>■ Use <a href="#">CsvImportTask.submit()</a> to submit the import task to the NetSuite task queue.</li> <li>■ Use the properties for the <a href="#">task.CsvImportTaskStatus</a> object to get the status of the import process.</li> </ul> <p>Use the following guidelines with the <b>CsvImportTask</b> Object:</p> <ul style="list-style-type: none"> <li>■ CSV imports performed within scripts are subject to the existing application limit of 25,000 records.</li> <li>■ 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.</li> <li>■ 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.</li> </ul> <p>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 <b>CsvImportTask</b>. When this object submits a CSV import job based on the saved import map, a single thread and single queue are used.</p>
---------------------------	---

	For a complete list of this object's methods and properties, see <a href="#">CsvImportTask Object Members</a> .
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//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()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Directs NetSuite to place a CSV import task into the NetSuite task queue and returns a unique ID for the task. Use <a href="#">CsvImportTaskStatus.status</a> 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.
<b>Returns</b>	string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	100 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
FAILED_TO_SUBMIT_JOB_REQUEST_1	Failed to submit job request: {reason}	Task cannot be submitted.

## Syntax

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

```
//Add additional code
...
var csvImportTaskId = csvTask.submit();
...
//Add additional code
```

## CsvImportTask.importFile

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	CSV file to import. Use a <a href="#">file.File</a> object or a string that represents the CSV text to be imported.
<b>Type</b>	<a href="#">file.File</a>   string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//Add additional code
...
var f = file.load('SuiteScripts/custjoblist.csv');
scriptTask.importFile = f;
...
//Add additional code
```

## CsvImportTask.mappingId

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Script ID or internal ID of the saved import map that you created when you ran the Import Assistant. See <a href="#">task.CsvImportTask</a> .
<b>Type</b>	number   string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>

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 task Module Script Sample.

```
//Add additional code
...
scriptTask.mappingId = 51;
...
//Add additional code
```

## CsvImportTask.queueId

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Overrides the <b>Queue Number</b> property under <b>Advanced Options</b> on the <b>Import Options</b> page of the Import Assistant. Use this property to programmatically select an import queue and improve performance during the import.
	<p><b>Note:</b> 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 <a href="#">Queue Number</a> and <a href="#">Use Multiple Threads and Multiple Queues to Run CSV Import Jobs</a>.</p>
<b>Type</b>	number
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//Add additional code
...
scriptTask.queueId = 2;
...
//Add additional code
```

## CsvImportTask.name

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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.
<b>Type</b>	string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//Add additional code
...
csvTask.name = 'Import Entities'
...
//Add additional code
```

## CsvImportTask.linkedFiles

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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 <a href="#">file.File</a> object or the raw CSV data to import.  You can assign multiple types of values to the <code>linkedFiles</code> property.
<b>Type</b>	Object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//Add additional code
...
scriptTask.linkedFiles = {'addressbook': 'street,city\nval1,val2', 'purchases': file.load('SuiteScripts/other.csv')};
...
//Add additional code
```

## task.CsvImportTaskStatus

<p><b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.</p>	
<b>Object Description</b>	Encapsulates the status of a CSV import task placed into the NetSuite scheduling queue.  Use <a href="#">task.checkStatus(options)</a> with the unique ID for the CSV import task to get the <b>CsvImportTaskStatus</b> object.  For a complete list of this object's properties, see <a href="#">CsvImportTaskStatus Object Members</a> .
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//Add additional code
...
var csvTaskStatus = task.checkStatus({
    taskId: csvTaskId
});
if (csvTaskStatus.status === task.TaskStatus.FAILED)
...
//Add additional code
```

## CsvImportTaskStatus.status

<p><b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.</p>	
<b>Property Description</b>	Status for a CSV import task. Returns a <a href="#">task.TaskStatus</a> enum value.
<b>Type</b>	<a href="#">task.TaskStatus</a>
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2015.2

### Errors

Error Code	Message	Thrown If
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 task Module Script Sample.

```
//Add additional code
...
var summary = task.checkStatus({
    taskId: scriptTaskId
});
log.audit({
    title: 'Status',
    details: summary.status
});
...
//Add additional code
```

## task.EntityDeduplicationTask

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	<p>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.</p> <p>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 &gt; Mass Update &gt; Entity Duplicate Resolution .</p> <p>For more information about merging duplicate records in NetSuite, see the help topic <a href="#">Merging or Deleting Duplicate Records</a>.</p> <p>To use the <b>EntityDeduplicationTask</b> Object:</p> <ul style="list-style-type: none"> <li>▪ Use <a href="#">task.create(options)</a> to create the <b>EntityDeduplicationTask</b> object.</li> <li>▪ Use <a href="#">EntityDeduplicationTask.entityType</a> to select the entity type on which you want to merge duplicate records.</li> <li>▪ Use <a href="#">EntityDeduplicationTask.dedupeMode</a> to select the action to take for the duplicate records.</li> <li>▪ Use a <a href="#">EntityDeduplicationTask.masterSelectionMode</a> enum value to identify which record to use as the master record in the merge.</li> <li>▪ If you use <a href="#">MasterSelectionMode.SELECT_BY_ID</a> for the master selection mode, set the ID of the master record with <a href="#">EntityDeduplicationTask.masterRecordId</a>.</li> <li>▪ Identify the duplicate records. Use the <a href="#">search.duplicates(options)</a> method in the <a href="#">N/search Module</a> to find the duplicate records.</li> <li>▪ Use <a href="#">EntityDeduplicationTask.submit()</a> to submit the merge duplicate record task to the NetSuite task queue.</li> <li>▪ Use the properties for the <a href="#">task.EntityDeduplicationTaskStatus</a> object to get the status of the merge duplicate record task.</li> </ul> <p>Use the following guidelines with the <b>EntityDeduplicationTask</b> Object:</p> <ul style="list-style-type: none"> <li>▪ You can only submit 200 records in a single merge duplicate records task.</li> <li>▪ The merge duplicate functionality on non-entity records is not supported in SuiteScript.</li> </ul>
---------------------------	--

	<ul style="list-style-type: none"> <li>■ You must have full access to the Duplicate Record Management permission to merge duplicates.</li> </ul> <p>For a complete list of this object's methods and properties, see <a href="#">EntityDeduplicationTask Object Members</a>.</p>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//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()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Directs NetSuite to place the merge duplicate records task into the NetSuite task queue and returns a unique ID for the task.  Use <a href="#">EntityDeduplicationTaskStatus.status</a> to view the status of a submitted task.
<b>Returns</b>	task id as a string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	100 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
FAILED_TO_SUBMIT_JOB_REQUEST_1	Failed to submit job request: {reason}	Task cannot be submitted.

## Syntax



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

```
//Add additional code
...
var dedupeTaskId = dedupeTask.submit();
...
//Add additional code
```

## EntityDeduplicationTask.entityType



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Sets the type of entity on which you want to merge duplicate records. Use a <a href="#">task.DedupeEntityType</a> enum value to set the value.
<b>Type</b>	<a href="#">task.DedupeEntityType</a>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//Add additional code
...
dedupeTask.entityType = task.DedupeEntityType.CUSTOMER;
...
//Add additional code
```

## EntityDeduplicationTask.masterRecordId



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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.
-----------------------------	--

	 <b>Important:</b> You must also select SELECT_BY_ID for the <code>EntityDeduplicationTask.masterSelectionMode</code> property, or NetSuite ignores this setting.
<b>Type</b>	number
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//Add additional code
...
dedupeTask.masterSelectionMode = task.MasterSelectionMode.SELECT_BY_ID;
dedupeTask.masterRecordId = 107;
...
//Add additional code
```

## EntityDeduplicationTask.masterSelectionMode

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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 <code>EntityDeduplicationTask.masterSelectionMode</code> to set the value.
<b>Type</b>	<code>task.MasterSelectionMode</code>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//Add additional code
```

```

...
dedupeTask.masterSelectionMode = task.MasterSelectionMode.MOST_RECENT_ACTIVITY;
...
//Add additional code

```

## EntityDeduplicationTask.dedupeMode



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Sets the mode in which to merge or delete duplicate records. Use a <a href="#">EntityDeduplicationTask.dedupeMode</a> enum value to set the value.
<b>Type</b>	<a href="#">task.DedupeMode</a>
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```

//Add additional code
...
dedupeTask.dedupeMode = task.DedupeMode.MERGE;
...
//Add additional code

```

## EntityDeduplicationTask.recordIds



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Number array of record internal IDs to perform the merge or delete operation on.  You can use the <a href="#">search.duplicates(options)</a> method to identify duplicate records or create an array with record internal IDs.
<b>Type</b>	number[]
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//Add additional code
...
dedupeTask.recordIds = ['107', '110'];
...
//Add additional code
```

## task.EntityDeduplicationTaskStatus

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the status of a merge duplicate record task placed into the NetSuite task queue by <a href="#">EntityDeduplicationTask.submit()</a> .  Use <a href="#">task.checkStatus(options)</a> 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> .
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//Add additional code
...
var dedupeTaskStatus = task.checkStatus({
    taskId: taskId
});
if (dedupeTaskStatus.status === task.TaskStatus.FAILED)
...
//Add additional code
```

## EntityDeduplicationTaskStatus.status

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Status for a merge duplicate record task. Returns a <a href="#">task.TaskStatus</a> enum value.
-----------------------------	---

Type	task.TaskStatus
Supported Script Types	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Module	N/task Module
Since	2015.2

## Errors

Error Code	Message	Thrown If
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 [task Module Script Sample](#).

```
//Add additional code
...
var summary = task.checkStatus({
    taskId: scriptTaskId
});
log.audit({
    title: 'Status',
    details: summary.status
});
...
//Add additional code
```

## task.SearchTask



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	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 <code>task.SearchTask</code> object using <a href="#">task.create(options)</a> .  Use the <code>task.SearchTask</code> object to do the following: <ul style="list-style-type: none"><li>■ Set the search ID using the <code>SearchTask.savedSearchId</code> property.</li><li>■ Set the file ID or file path of a CSV file in the File Cabinet. Search results are exported to this file. Use the <code>SearchTask fileId</code> property or the <code>SearchTask.filePath</code> property. Exactly one of these properties must be set. If both are set, an error occurs.</li><li>■ Add dependent scripts to the search task using <code>SearchTask.addInboundDependency()</code>. Dependent scripts are processed automatically when the search task is complete.</li><li>■ Submit the search task to the NetSuite task queue using <code>SearchTask.submit()</code>.</li><li>■ Get the status of a search task using the properties of the <code>task.SearchTaskStatus</code> object.</li></ul>
---------------------------	--

	<p><b>Note:</b> There is 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.</p>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2017.1

## Syntax

```
//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()

<p><b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.</p>	
<b>Method Description</b>	Adds a scheduled script task ( <a href="#">task.ScheduledScriptTask</a> ) or map/reduce script task ( <a href="#">task.MapReduceScriptTask</a> ) 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 <a href="#">SuiteCloud Processors</a> .
	<p><b>Note:</b> You can add only scheduled scripts or map/reduce scripts as dependent scripts to asynchronous search tasks. Other script types are not supported.</p>
	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 to a search task as a dependent script, the scheduled script depends on the search task. Because <code>addInboundDependency()</code> is called on the search task, any dependent scripts that you add are considered inbound dependencies.
<b>Returns</b>	void
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2018.2

## Parameters

Parameter	Type	Required / Optional	Description	Since
dependentScript	task.ScheduledScriptTask   task.MapReduceScriptTask	Required	<p>The script to add as a dependent script to the search task.</p> <p>Use <code>task.create(options)</code> and the <code>task.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.</p> <p>You can add only one dependent script per call to <code>SearchTask.addInboundDependency()</code>.</p>	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 [task Module Script Sample](#).

```
// 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 asyncTask = task.create({
    taskType: task.TaskType.SEARCH
});
// Set the properties of the search task
asyncTask.savedSearchId = 'customsearch35';
...

asyncTask.addInboundDependency(scheduledScript);

var asyncTaskId = asyncTask.submit();
...
// Add additional code
```

## SearchTask.submit()

**ℹ Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">SearchTask.addInboundDependency()</a> ) to the <a href="#">SearchTask.inboundDependencies</a> property.  Use <a href="#">task.SearchTaskStatus</a> to view the status of a submitted task.
<b>Returns</b>	The task ID as a string
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	100 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2017.1

## Errors

Error Code	Thrown If
FAILED_TO_SUBMIT_JOB_REQUEST_1	Task cannot be submitted.
SSS_MISSING_REQD_ARGUMENT	A required parameter is missing.
YOU_DO_NOT_HAVE_ACCESS_TO_THE_MEDIA_ITEM_YOU_SELECTED	You do not have permission to access the file.
THAT_RECORD_DOES_NOT_EXIST	The file Object references a file that doesn't exist.
MUST_IDENTIFY_A_FILE	The path specifies a folder and not a file.
CANNOT_RESUBMIT_SUBMITTED_ASYNC_SEARCH_TASK	The search task was already submitted and completed successfully.
ASYNC_SEARCH_DEPENDENCY_MR_ALREADY_SUBMITTED	A dependent map/reduce script is already submitted and is not complete.
ASYNC_SEARCH_DEPENDENCY_MR_INCORRECT_STATUS	The status of the deployment record for the specified dependent map/reduce script has a value other than "Not Scheduled".
ASYNC_SEARCH_DEPENDENCY_SS_ALREADY_SUBMITTED	A dependent scheduled script is already submitted and is not complete.
ASYNC_SEARCH_DEPENDENCY_SS_INCORRECT_STATUS	The status of the deployment record for the specified dependent scheduled script has a value other than "Not Scheduled".
ASYNC_SEARCH_DEPLOYMENT_FOR_DEPENDENCY	A deployment record for the specified dependent script is not available for one of the following reasons: <ul style="list-style-type: none"> <li>■ A deployment record was not specified when the dependent script was created, and automatic lookup for an available deployment record failed.</li> <li>■ The deployment record specified when the dependent script was created is not found.</li> </ul>
ASYNC_SEARCH_MULTIPLE_DEPENDENCIES	The same dependent script is passed to this method more than once.
ASYNC_SEARCH_SCRIPT_ID_NOT_FOUND	The specified dependent script is not found.

Error Code	Thrown If
ASYNC_SEARCH_SEARCH_ID_NOT_FOUND	The search task with the specified search ID is not found.

## Syntax

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

```
//Add additional code
...
var searchTriggerTask = searchTask.submit();
...
//Add additional code
```

## SearchTask.fileId

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	ID of the CSV file to export search results into.
	<p><b>Note:</b> Either this property or the <a href="#">SearchTask.filePath</a> property must be set. If both are set, an error occurs.</p>
<b>Type</b>	The CSV file ID as a number
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2017.1

## Errors

Error Code	Thrown If
UNSUPPORTED_COMBINATION_OF_PARAMETERS	Both this property and the <a href="#">SearchTask.filePath</a> property are set at the same time.

## Syntax

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

```
//Add additional code
...
searchTask.fileId = 18;
...
//Add additional code
```

## SearchTask.filePath

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Path of the CSV file to export search results into.   <b>Note:</b> Either this property or the <a href="#">SearchTask.fileId</a> property must be set. If both are set, an error occurs.
<b>Type</b>	The CSV file path as a string
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2017.1

### Errors

Error Code	Thrown If
UNSUPPORTED_COMBINATION_OF_PARAMETERS	Both this property and the <a href="#">SearchTask.fileId</a> property are set at the same time.

### Syntax

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

```
//Add additional code
...
searchTask.filePath= 'ExportFolder/export.csv'
...
//Add additional code
```

## SearchTask.inboundDependencies

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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 <a href="#">SearchTask.addInboundDependency()</a> .  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:

```
{"0": {"type": "task.ScheduledScriptTask", "scriptId": "customscript_as_ftr_ss", "deploymentId": "customdeploy_ss_dpl", "params": {"custscript_ss_as_srch_res": "SuiteScripts/ExportFile.csv"}}, "1": {"type": "task.MapReduceScriptTask", "scriptId": "customscript_as_ftr_mr", "deploymentId": "customdeploy_mr_dpl", "params": {"custscript_mr_as_srch_res": "SuiteScripts/ExportFile.csv"}}}
```

After you submit the search task, the internal IDs of the dependent scripts are added to the `SearchTask.inboundDependencies` property:

```
{"0": {"type": "task.ScheduledScriptTask", "id": "SCHEDSCRIPT_0168697b126d1705061d0d690a787755500b046a1912686b10_349d94266564827c739a2ba0a5b9d476f4097217", "scriptId": "customscript_as_ftr_ss", "deploymentId": "customdeploy_ss_dpl", "params": {"custscript_ss_as_srch_res": "SuiteScripts/ExportFile.csv"}}, "1": {"type": "task.MapReduceScriptTask", "id": "MAPREDUCETASK_0268697b126d1705061d0d69027f7b39560f01001c_7a02acb4bdebff0103120b09302170720aa57bca4", "scriptId": "customscript_as_ftr_mr", "deploymentId": "customdeploy_mr_dpl", "params": {"custscript_mr_as_srch_res": "SuiteScripts/ExportFile.csv"}}}
```

Type	read-only Object[]
Supported Script Types	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Module	N/task Module
Since	2018.2

## Errors

Error Code	Thrown If
READ_ONLY_PROPERTY	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 [task Module Script Sample](#).

```
// 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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	ID of the saved search to be executed during the task.
<b>Type</b>	The saved search ID as a number
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```

//Add additional code
...
searchTask.savedSearchId = 51;
...
//Add additional code

```

## task.SearchTaskStatus



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the status of an asynchronous search task ( <a href="#">task.SearchTask</a> ) placed into the NetSuite task queue. To initiate the task and retrieve the task ID, use <a href="#">SearchTask.submit()</a> .
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
// Add additional code
...
var searchTaskStatus = task.checkStatus({
    taskId: 51
});

if (searchTaskStatus.status === task.TaskStatus.FAILED) {
    // Handle the task failure
}
...
// Add additional code
```

## SearchTaskStatus.fileId



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	ID of CSV file into which search results are exported.
<b>Type</b>	CSV file id as a number
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2017.1

### Errors

Error Code	Message	Thrown If
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 task Module Script Sample.

```
//Add additional code
...
var status = task.checkStatus({
    searchTaskId: 81
});
log.audit({
    title: 'File ID',
    details: status.fileId
});
...
//Add additional code
```

## SearchTaskStatus.savedSearchId

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The ID of the saved search executed during the task.
<b>Type</b>	The search ID as a number
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2017.1

## Errors

Error Code	Message	Thrown If
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 task Module Script Sample.

```
//Add additional code
...
var status = task.checkStatus({
    searchTaskId: 81
});
log.audit({
    title: 'Saved Search ID',
    details: status.savedSearchId
});
...
//Add additional code
```

## SearchTaskStatus.status

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Status for an asynchronous search placed in the NetSuite task queue by <code>SearchTask.submit()</code> . Returns a <code>task.TaskStatus</code> enum value.
<b>Type</b>	<code>task.TaskStatus</code>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2017.1

### Errors

Error Code	Message	Thrown If
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 [task Module Script Sample](#).

```
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
log.audit({
    title: 'Status',
    details: summary.status
});

...
//Add additional code
```

## SearchTaskStatus.taskId

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	ID of the <code>task.SearchTask</code> Object. Use <code>SearchTask.submit()</code> to return this ID.
<b>Type</b>	<code>number</code>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2017.1

## Errors

Error Code	Message	Thrown If
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 task Module Script Sample.

```
//Add additional code
...
var searchTaskId = searchTask.submit();
...
//Add additional code
```

## task.WorkflowTriggerTask

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	<p>Encapsulates all the properties required to asynchronously initiate a workflow. Use the <b>WorkflowTriggerTask</b> Object to create a task that initiates an instance of the specified workflow.</p> <p>The task is placed in the scheduling queue, and the workflow instance is initiated after the task reaches the top of the queue.</p> <p><b>task.WorkflowTriggerTask</b> does not successfully place a workflow job in queue if an identical instance of that workflow (with the same <b>recordType</b>, <b>id</b>, and <b>workflowId</b>) is currently executing or already in the scheduling queue.</p> <p>To use the <b>WorkflowTriggerTask</b> Object:</p> <ul style="list-style-type: none"> <li>■ Use <a href="#">task.create(options)</a> to create the <b>WorkflowTriggerTask</b> Object.</li> <li>■ Use <a href="#">WorkflowTriggerTask.recordType</a> to set the record type of the workflow base record.</li> <li>■ Use <a href="#">WorkflowTriggerTask.recordId</a> to set the internal ID of the base record for the workflow.</li> <li>■ Use <a href="#">WorkflowTriggerTask.workflowId</a> to set the internal ID of the workflow that you want to run on the record specified by the <b>recordId</b>.</li> <li>■ Optionally, use <a href="#">WorkflowTriggerTask.params</a> to specify default values for workflow fields.</li> <li>■ Use <a href="#">WorkflowTriggerTask.submit()</a> to submit the asynchronous workflow initiation task to the NetSuite task queue.</li> <li>■ Use the properties for the <a href="#">WorkflowTriggerTaskStatus.status</a> object to get the status of the workflow execution.</li> </ul> <p>Use the following guidelines with the <b>WorkflowTriggerTask</b> Object:</p> <ul style="list-style-type: none"> <li>■ <a href="#">WorkflowTriggerTask.submit()</a> does not successfully place a workflow task in the scheduling queue if an identical instance of that workflow, with the same <b>recordType</b>, <b>recordId</b>, and <b>workflowId</b>, is currently executing or already in the scheduling queue.</li> </ul> <p>For a complete list of this object's methods and properties, see <a href="#">WorkflowTriggerTask Object Members</a>.</p> <p>To initiate a workflow on demand, see <a href="#">workflow.initiate(options)</a>.</p>
---------------------------	--

<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//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()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Directs NetSuite to place the asynchronous workflow initiation task into the NetSuite scheduling queue and returns a unique ID for the task.  Use <a href="#">WorkflowTriggerTaskStatus.status</a> to view the status of a submitted task.
<b>Returns</b>	the task id as a string
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	20 units
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
FAILED_TO_SUBMIT_JOB_REQUEST_1	Failed to submit job request: {reason}	Task cannot be submitted.

## Syntax

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

```
//Add additional code
```

```

...
var workflowTriggerTask = workflowTask.submit();
...
//Add additional code

```

## WorkflowTriggerTask.recordType

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Record type of the workflow definition base record. For example, customer, salesorder, or lead.  In the Workflow Manager, this is the record type that is specified in the Record Type field.
<b>Type</b>	string
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 [task Module Script Sample](#).

```

//Add additional code
...
workflowTask.recordType = 'customer';
...
//Add additional code

```

## WorkflowTriggerTask.recordId

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Internal ID of the base record. For example, 55 or 124.
<b>Type</b>	number
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//Add additional code
...
workflowTask.recordId = 107;
...
//Add additional code
```

## WorkflowTriggerTask.workflowId

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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 <a href="#">Workflow Definition Page</a> .
<b>Type</b>	number   string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//Add additional code
...
workflowTask.workflowId = 3;
...
//Add additional code
```

## WorkflowTriggerTask.params

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Object that contains key/value pairs to set default values on fields specific to the workflow.  These can include fields on the <a href="#">Workflow Definition Page</a> or workflow and state <a href="#">Workflow Custom Fields</a> .
<b>Type</b>	Object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Module</b>	N/task Module
<b>Since</b>	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 task Module Script Sample.

```
//Add additional code
...
task.params = {context: portlet}
...
//Add additional code
```

## task.WorkflowTriggerTaskStatus

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the status of an asynchronous workflow initiation task placed into the NetSuite task queue by <a href="#">WorkflowTriggerTask.submit()</a> .  Use <a href="#">task.checkStatus(options)</a> 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 <a href="#">WorkflowTriggerTaskStatus Object Members</a> .
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	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 task Module Script Sample.

```
//Add additional code
...
var workflowTaskStatus = task.checkStatus(taskId);
if (workflowTaskStatus.status === task.TaskStatus.FAILED)
...
//Add additional code
```

## WorkflowTriggerTaskStatus.status

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Status for an asynchronous workflow placed in the NetSuite task queue by <a href="#">WorkflowTriggerTask.submit()</a> . Returns a <a href="#">task.TaskStatus</a> enum value.
-----------------------------	---

Type	task.TaskStatus
Supported Script Types	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Module	<a href="#">N/task Module</a>
Since	2015.2

## Errors

Error Code	Message	Thrown If
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 [task Module Script Sample](#).

```
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
log.audit('Status', summary.status);

...
//Add additional code
```

## task.RecordActionTask

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	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.
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Methods and Properties</b>	<a href="#">RecordActionTask Object Members</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//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 content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <code>task.checkStatus()</code> method.
<b>Returns</b>	string
<b>Supported Script Types</b>	Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	50 units
<b>Module</b>	N/task Module
<b>Sibling Object Members</b>	<a href="#">RecordActionTask Object Members</a>
<b>Since</b>	2019.1

### Errors

Error Code	Message	Thrown If
FAILED_TO_SUBMIT_JOB_REQUEST_1	Failed to submit job request: {reason}	The task cannot be submitted.

### Syntax

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

```

//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()

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the object type name.
<b>Returns</b>	string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/task Module
<b>Sibling Object Members</b>	<a href="#">RecordActionTask Object Members</a>
<b>Since</b>	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 [task Module Script Sample](#).

```

// Add additional code  
  

var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});  

log.debug("Task type: " + recordActionTask.toString());  
  

// Add additional code

```

## RecordActionTask.toJSON()

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns an object in JSON.
<b>Returns</b>	Object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Governance</b>	None
<b>Module</b>	N/task Module
<b>Sibling Object Members</b>	RecordActionTask Object Members
<b>Since</b>	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 task Module Script Sample.

```
// 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 content in this help topic pertains to SuiteScript 2.0.

<b>Description</b>	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 <a href="#">task.ActionCondition</a> . This parameter cannot be specified when <a href="#">RecordActionTask.params</a> is specified.
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/task Module
<b>Sibling Object Members</b>	RecordActionTask Object Members
<b>Since</b>	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 task Module Script Sample.

```
// 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 content in this help topic pertains to SuiteScript 2.0.

<b>Parameter Description</b>	The record type on which the action is to be performed. For a list of record types, see <a href="#">record.Type</a> .
<b>Type</b>	string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/task Module
<b>Sibling Object Members</b>	<a href="#">RecordActionTask Object Members</a>
<b>Since</b>	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 [task Module Script Sample](#).

```

//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 content in this help topic pertains to SuiteScript 2.0.

<b>Parameter Description</b>	The ID of the action to be invoked.
------------------------------	-------------------------------------

Type	string
Supported Script Types	<p>Server scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
Module	<a href="#">N/task Module</a>
Sibling Object Members	<a href="#">RecordActionTask Object Members</a>
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 [task Module Script Sample](#).

```
//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 content in this help topic pertains to SuiteScript 2.0.

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	<p>Server scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
Module	<a href="#">N/task Module</a>
Sibling Object Members	<a href="#">RecordActionTask Object Members</a>
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 task Module Script Sample.

```
//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.condition

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Parameter Description</b>	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: <code>function(v) { return { recordId: v }; }.</code>
<b>Type</b>	Object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Sibling Object Members</b>	<a href="#">RecordActionTask Object Members</a>
<b>Since</b>	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 task Module Script Sample.

```
// 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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	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.
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Methods and Properties</b>	<a href="#">RecordActionTaskStatus Object Members</a>
<b>Since</b>	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 [task Module Script Sample](#).

```

//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: [] },
            5: { response: { approvedId: 5 }, notifications: [{ title: 'Title', message: 'Message', severity:
{ value: 2, label: 'Warning' }}] },
            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: [] },
            5: { response: { approvedId: 5 }, notifications: [{ title: 'Title', message: 'Message', severity:
{ value: 2, label: 'Warning' }}] }
        },
        errors: {
            23: { name: 'SSS_RECORD_DOES_NOT_SATISFY_CONDITION', message: ... }
        },
        complete: 3,
        succeeded: 2,
        failed: 1,
        pending: 0
    }
}

...

```

```
//Add additional code
```

## RecordActionTaskStatus.toString()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the object type name.
<b>Returns</b>	string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/task Module
<b>Sibling Object Members</b>	<a href="#">RecordActionTaskStatus Object Members</a>
<b>Since</b>	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 [task Module Script Sample](#).

```
//Add additional code
...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.params = [{recorderId: 1}, {recorderId: 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 content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a record status task status object in JSON.
<b>Returns</b>	Object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/task Module

<b>Sibling Object Members</b>	RecordActionTaskStatus Object Members
<b>Since</b>	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 task Module Script Sample.

```
//Add additional code
...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.params = [{recorderId: 1}, {recorderId: 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 content in this help topic pertains to SuiteScript 2.0.

<b>Parameter Description</b>	Represents the record action task status. Returns a value from the <code>task.TaskStatus</code> enum.
<b>Type</b>	string
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/task Module
<b>Sibling Object Members</b>	RecordActionTaskStatus Object Members
<b>Since</b>	2019.1

## Errors

Error Code	Thrown If
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 task Module Script Sample.

```
// Add additional code
```

```

...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.params = [{recorderId: 1}, {recorderId: 2}];
var handle = recordActionTask.submit();

var taskStatus = task.checkStatus({taskId: 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 content in this help topic pertains to SuiteScript 2.0.

<b>Parameter Description</b>	The results of successfully executed record action tasks. The value of the property is the task instance ID and the corresponding action result.
<b>Type</b>	Object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Sibling Object Members</b>	<a href="#">RecordActionTaskStatus Object Members</a>
<b>Since</b>	2019.1

### Errors

Error Code	Thrown If
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 [task Module Script Sample](#).

```

// Add additional code
...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.params = [{recorderId: 1}, {recorderId: 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 content in this help topic pertains to SuiteScript 2.0.

<b>Parameter Description</b>	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.
<b>Type</b>	Object
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Sibling Object Members</b>	<a href="#">RecordActionTaskStatus Object Members</a>
<b>Since</b>	2019.1

### Errors

Error Code	Thrown If
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 [task Module Script Sample](#).

```

// 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 content in this help topic pertains to SuiteScript 2.0.

<b>Parameter Description</b>	The number of record actions that are already executed, either failed or successful.
<b>Type</b>	number
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/task Module
<b>Sibling Object Members</b>	<a href="#">RecordActionTaskStatus Object Members</a>
<b>Since</b>	2019.1

### Errors

Error Code	Thrown If
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 [task Module Script Sample](#).

```
// 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 content in this help topic pertains to SuiteScript 2.0.

<b>Parameter Description</b>	The number of record actions with a successful status.
<b>Type</b>	number
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Module</b>	N/task Module
<b>Sibling Object Members</b>	RecordActionTaskStatus Object Members
<b>Since</b>	2019.1

## Errors

Error Code	Thrown If
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 task Module Script Sample.

```
// 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 content in this help topic pertains to SuiteScript 2.0.

<b>Parameter Description</b>	The number of record actions with a failed status.
<b>Type</b>	number
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/task Module
<b>Sibling Object Members</b>	RecordActionTaskStatus Object Members
<b>Since</b>	2019.1

## Errors

Error Code	Thrown If
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 task Module Script Sample.

```
// 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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

Parameter Description	The number of record actions with a pending status.
Type	number
Supported Script Types	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Module	<a href="#">N/task Module</a>
Sibling Object Members	<a href="#">RecordActionTaskStatus Object Members</a>
Since	2019.1

## Errors

Error Code	Thrown If
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 task Module Script Sample.

```
// 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.ActionCondition

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for the possible record action conditions. This enum is returned by <a href="#">RecordActionTask.condition</a> .
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2019.1

### 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 [task Module Script Sample](#).

```

// 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.create(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates an object for a specific task type and returns the task object. Use with the <a href="#">N/task Module</a> to create a task to schedule scripts, run map/reduce scripts, import CSV files, merge duplicate records, initiate asynchronous searches, or execute asynchronous workflows.
<b>Returns</b>	<a href="#">task.ScheduledScriptTask</a>   <a href="#">task.MapReduceScriptTask</a>   <a href="#">task.CsvImportTask</a>   <a href="#">task.EntityDeduplicationTask</a>   <a href="#">task.WorkflowTriggerTask</a>   <a href="#">task.SearchTask</a>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.taskType</code>	<a href="#">task.TaskType</a>	Required	The type of task object to create. Use the <a href="#">task.TaskType</a> enum to set the value.	2015.2
<code>options.scriptId</code>	number   string	Optional	The internal ID (as a number) or script ID (as a string) for the script record.  This parameter sets the value for the <a href="#">ScheduledScriptTask.scriptId</a> or <a href="#">MapReduceScriptTask.scriptId</a> property.  Only applicable when <code>taskType</code> is set to <b>SCHEDED_SCRIPT</b> or <b>MAP_REDUCE</b> .	2016.2
<code>options.deploymentId</code>	string	Optional	The script ID (as a string) of the script deployment record.  This parameter sets the value for the <a href="#">ScheduledScriptTask.deploymentId</a> or <a href="#">MapReduceScriptTask.deploymentId</a> property.  Only applicable when <code>taskType</code> is set to <b>SCHEDED_SCRIPT</b> or <b>MAP_REDUCE</b> .	2016.2
<code>options.params</code>	Object	Optional	An object that represents key/value pairs that override static script parameter field values on the script deployment record.  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 <a href="#">Creating Script Parameters Overview</a> .  For Workflow tasks, keys can include fields on the <a href="#">Workflow Definition Page</a> or workflow and state <a href="#">Workflow Custom Fields</a> .	2016.2

Parameter	Type	Required / Optional	Description	Since
			<p>This parameter sets the value for the <a href="#">ScheduledScriptTask.params</a>, <a href="#">MapReduceScriptTask.params</a> or <a href="#">WorkflowTriggerTask.params</a> property.</p> <p>Only applicable when <code>taskType</code> is set to <b>SCHEDULED_SCRIPT</b>, <b>MAP_REDUCE</b> or <b>WORKFLOW_TRIGGER</b>.</p>	
<code>options.importFile</code>	<code>file.File</code>   string	Optional	<p>A CSV file to import. Use a <code>file.File</code> object or a string that represents the CSV text to be imported.</p> <p>This parameter sets the value for the <a href="#">CsvImportTask.importFile</a> property.</p> <p>Only applicable when <code>taskType</code> is set to <b>CSV_IMPORT</b>.</p>	2016.2
<code>options.mappingId</code>	number   string	Optional	<p>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 <a href="#">task.CsvImportTask</a>.</p> <p>This parameter sets the value for the <a href="#">CsvImportTask.mappingId</a> property.</p> <p>Only applicable when <code>taskType</code> is set to <b>CSV_IMPORT</b>.</p>	2016.2
<code>options.queueId</code>	number	Optional	<p>Overrides the <b>Queue Number</b> property under <b>Advanced Options</b> on the <b>Import Options</b> page of the Import Assistant. Use this property to programmatically select an import queue and improve performance during the import.</p> <p><b>Note:</b> 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 <a href="#">Queue Number</a> and <a href="#">Use Multiple Threads and Multiple Queues to Run CSV Import Jobs</a>.</p> <p>This parameter sets the value for the <a href="#">CsvImportTask.queueId</a> property.</p> <p>Only applicable when <code>taskType</code> is set to <b>CSV_IMPORT</b>.</p>	2016.2
<code>options.name</code>	string	Optional	<p>The name for the CSV import task.</p> <p>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.</p> <p>This parameter sets the value for the <a href="#">CsvImportTask.name</a> property.</p> <p>Only applicable when <code>taskType</code> is set to <b>CSV_IMPORT</b>.</p>	2016.2
<code>options.linkedFiles</code>	Object	Optional	<p>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.</p> <p>The key is the internal ID of the record sublist for which data is being imported and the value is either a <code>file.File</code> object or the raw CSV data to import.</p>	2016.2

Parameter	Type	Required / Optional	Description	Since
			<p>You can assign multiple types of values to the <code>linkedFiles</code> property.</p> <p>This parameter sets the value for the <code>CsvImportTask.linkedFiles</code> property.</p> <p>Only applicable when <code>taskType</code> is set to <b>CSV_IMPORT</b>.</p>	
<code>options.entityType</code>	string	Optional	<p>Sets the type of entity on which you want to merge duplicate records.</p> <p>This parameter sets the value for the <code>EntityDeduplicationTask.entityType</code> property.</p> <p>Only applicable when <code>taskType</code> is set to <b>ENTITY_DEDUPLICATION</b>.</p> <p>Use the <code>task.DedupeEntityType</code> enum to set the value.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <span style="color: #0070C0; font-size: 1.5em; border-radius: 50%; padding: 2px 5px; margin-right: 10px;"></span> <b>Note:</b> If you set <code>entityType</code> to CUSTOMER, the system will automatically include prospects and leads in the task request.         </div>	2016.2
<code>options.masterRecordId</code>	number	Optional	<p>When you merge duplicate records, you can delete all duplicates for a record or merge information from the duplicate records into the master record.</p> <p>Use this property to set the ID of the master record that you want to use as the master record in the merge.</p> <div style="border: 1px solid #ccc; background-color: #FFFACD; padding: 10px; margin-top: 10px;"> <span style="color: #FF8C00; font-size: 1.5em; border-radius: 50%; padding: 2px 5px; margin-right: 10px;"></span> <b>Important:</b> You must also select <code>SELECT_BY_ID</code> for the <code>EntityDeduplicationTask.masterSelectionMode</code> property, or NetSuite ignores this setting.         </div> <p>This parameter sets the value for the <code>EntityDeduplicationTask.masterRecordId</code> property.</p> <p>Only applicable when <code>taskType</code> is set to <b>ENTITY_DEDUPLICATION</b>.</p>	2016.2
<code>options.masterSelectionMode</code>	string	Optional	<p>When you merge duplicate records, you can delete all duplicates for a record or merge information from the duplicate records into the master record.</p> <p>Set this property to determine which of the duplicate records to keep or select the master record to use by ID.</p> <p>This parameter sets the value for the <code>EntityDeduplicationTask.masterSelectionMode</code> property.</p> <p>Only applicable when <code>taskType</code> is set to <b>ENTITY_DEDUPLICATION</b>.</p> <p>Use the <code>task.MasterSelectionMode</code> enum to set the value.</p>	2016.2
<code>options.dedupeMode</code>	string	Optional	Sets the mode in which to merge or delete duplicate records.	2016.2

Parameter	Type	Required / Optional	Description	Since
			<p>This parameter sets the value for the <a href="#">EntityDeduplicationTask.dedupeMode</a> property.</p> <p>Only applicable when <code>taskType</code> is set to <b>ENTITY_DEDUPLICATION</b>.</p> <p>Use the <a href="#">task.DedupeMode</a> enum to set the value.</p>	
<code>options.recordIds</code>	number[]	Optional	<p>The number array of record internal IDs to perform the merge or delete operation on.</p> <p>You can use the <a href="#">search.duplicates(options)</a> method to identify duplicate records or create an array with record internal IDs.</p> <p>This parameter sets the value for the <a href="#">EntityDeduplicationTask.recordIds</a> property.</p> <p>Only applicable when <code>taskType</code> is set to <b>ENTITY_DEDUPLICATION</b>.</p>	2016.2
<code>options.recordType</code>	string	Optional	<p>The record type of the workflow definition base record, such as customer, salesorder, or lead.</p> <p>In the Workflow Manager, this is the record type that is specified in the Record Type field.</p> <p>This parameter sets the value for the <a href="#">WorkflowTriggerTask.recordType</a> property.</p> <p>Only applicable when <code>taskType</code> is set to <b>WORKFLOW_TRIGGER</b>.</p>	2016.2
<code>options.recordId</code>	number	Optional	<p>The internal ID of the base record.</p> <p>This parameter sets the value for the <a href="#">WorkflowTriggerTask.recordId</a> property.</p> <p>Only applicable when <code>taskType</code> is set to <b>WORKFLOW_TRIGGER</b>.</p>	2016.2
<code>options.workflowId</code>	number   string	Optional	<p>The internal ID (as a number) or script ID (as a string) for the workflow definition.</p> <p>This is the ID that appears in the ID field on the <a href="#">Workflow Definition Page</a>.</p> <p>This parameter sets the value for the <a href="#">WorkflowTriggerTask.workflowId</a> property.</p> <p>Only applicable when <code>taskType</code> is set to <b>WORKFLOW_TRIGGER</b>.</p>	2016.2
<code>options.savedSearchId</code>	number	Optional	The ID of the saved search to be executed during the task.	2017.1
<code>options.fileId</code>	string	Optional	<p>The ID of the CSV file to export search results to. See <a href="#">N/file Module</a>.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <span style="color: #0070C0; font-size: 1.5em; border-radius: 50%; width: 1.2em; height: 1.2em; display: inline-block; vertical-align: middle;"></span> <b>Note:</b> If <code>fileId</code> is provided then the <code>filePath</code> parameter is ignored. There is no synchronization between <code>fileId</code> and <code>filePath</code> values.         </div>	2017.1
<code>options.filePath</code>	number	Optional	Path of the CSV file to export search results to. See <a href="#">N/file Module</a> .	2017.1

Parameter	Type	Required / Optional	Description	Since
			<p><b>Note:</b> If <code>fileId</code> is provided then the <code>filePath</code> parameter is ignored. There is no synchronization between <code>fileId</code> and <code>filePath</code> values.</p>	

## Syntax

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

```
//Add additional code
...
var mrTask = task.create({
    taskType: task.TaskType.MAP_REDUCE,
    scriptId: 34,
    deploymentId: custdeploy1,
    params: {doSomething: true}
});
...
//Add additional code
```

## task.checkStatus(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a task status object associated with a specific task ID.
<b>Returns</b>	<code>task.ScheduledScriptTaskStatus</code>   <code>task.MapReduceScriptTaskStatus</code>   <code>task.CsvImportTaskStatus</code>   <code>task.EntityDeduplicationTaskStatus</code>   <code>task.SearchTaskStatus</code>   <code>task.WorkflowTriggerTaskStatus</code>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/task Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.taskId	<code>task.ScheduledScriptTask</code>   <code>task.MapReduceScriptTask</code>   <code>task.CsvImportTask</code>   <code>task.EntityDeduplicationTask</code>	Required	Unique ID for the task that was generated by <code>task.create(options)</code> .	2015.2

Parameter	Type	Required / Optional	Description	Since
	task.SearchTask task.WorkflowTriggerTask			

## Syntax

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

```
//Add additional code
...
var taskStatus = task.checkStatus(mrTaskId);
...
//Add additional code
```

## task.TaskType

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for the types of task objects supported by the <a href="#">N/task Module</a> , that you can create with <a href="#">task.create(options)</a> .
	<b>i Note:</b> 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.
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2015.2

## Values

- SCHEDULED\_SCRIPT
- MAP\_REDUCE
- CSV\_IMPORT
- ENTITY\_DEDUPLICATION
- SEARCH
- WORKFLOW\_TRIGGER
- RECORD\_ACTION

## Syntax

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

```
//Add additional code
```

```
...
var mrTask = task.create({
  taskType: task.TaskType.MAP_REDUCE
});

...
//Add additional code
```

## task.TaskStatus



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	<p>Enumeration that holds the string values for the possible status of tasks created and submitted with the <a href="#">N/task Module</a>.</p> <p>The following properties hold a value for <code>task.taskStatus</code>:</p> <ul style="list-style-type: none"> <li>▪ <a href="#">ScheduledScriptTaskStatus.status</a></li> <li>▪ <a href="#">MapReduceScriptTaskStatus.status</a></li> <li>▪ <a href="#">CsvImportTaskStatus.status</a></li> <li>▪ <a href="#">EntityDeduplicationTaskStatus.status</a></li> <li>▪ <a href="#">SearchTaskStatus.status</a></li> <li>▪ <a href="#">WorkflowTriggerTaskStatus.status</a></li> <li>▪ <a href="#">PiRemovalTaskStatus.status</a></li> <li>▪ <a href="#">PiRemovalTaskLogItem.status</a></li> </ul> <p><b>Note:</b> 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>
<b>Supported Script Types</b>	<p>Server scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2015.2

### 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 [task Module Script Sample](#).

```
//Add additional code
```

```

...
if (status == task.TaskStatus.COMPLETE || status == task.TaskStatus.FAILED)
...
//Add additional code

```

## task.MasterSelectionMode



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	<p>Enumeration that holds the string values for supported master selection modes when merging duplicate records with <a href="#">task.EntityDeduplicationTask</a>.</p> <p>Use this enum for the <a href="#">EntityDeduplicationTask.masterSelectionMode</a> property.</p> <p>For more information about these values, see the help topic <a href="#">Merging or Deleting Duplicate Records</a>.</p> <p><b>Note:</b> 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>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2015.2

### 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 [task Module Script Sample](#).

```

//Add additional code
...
dedupeTask.masterSelectionMode = task.MasterSelectionMode.MOST_RECENT_ACTIVITY;
...
//Add additional code

```

## task.DedupeMode



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for the available deduplication modes when merging duplicate records with <a href="#">task.EntityDeduplicationTask</a> . Use this enum for the <a href="#">EntityDeduplicationTask.dedupeMode</a> property.
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2015.2

### Values

- MERGE
- DELETE
- MAKE\_MASTER\_PARENT
- MARK\_AS\_NOT\_DUPES

### Syntax



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

```
//Add additional code
...
dedupeTask.dedupeMode = task.DedupeMode.MERGE;
...
//Add additional code
```

## task.DedupeEntityType



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for entity types for which you can merge duplicate records with <a href="#">task.EntityDeduplicationTask</a> . Use this enum for the <a href="#">EntityDeduplicationTask.entityType</a> .
-------------------------	---

	<p><b>Note:</b> 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>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task Module</a>
<b>Since</b>	2015.2

## 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 [task Module Script Sample](#).

```
//Add additional code
...
dedupeTask.entityType = task.DedupeEntityType.CUSTOMER;
...
//Add additional code
```

## task.MapReduceStage

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	<p>Enumeration that holds the string values for possible stages in <a href="#">task.MapReduceScriptTask</a> for a map/reduce script.</p> <p>This enum is returned by <a href="#">MapReduceScriptTaskStatus.stage</a>.</p> <p>For general information about map/reduce stages, see the help topics <a href="#">Map/Reduce Key Concepts</a> and <a href="#">SuiteScript 2.0 Map/Reduce Script Stages</a>.</p> <p><b>Note:</b> 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>
<b>Supported Script Types</b>	Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Module</b>	N/task Module
<b>Since</b>	2015.2

## 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 task Module Script Sample.

```
//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](#).

## N/task/accounting/recognition Module

**Note:** The content in this help topic pertains to SuiteScript 2.0.

Load the N/task/accounting/recognition module to merge revenue arrangements or revenue elements. A revenue arrangement is a transaction that records the details of a sale for the purposes of revenue allocation and recognition. The N/task/accounting/recognition module lets you combine revenue arrangements or revenue elements from multiple sources to represent a single contract obligation for revenue allocation and recognition.

You can use the [recognition.create\(options\)](#) method to create a merge task that combines entire revenue arrangements or individual revenue elements. This method returns a [recognition.MergeArrangementsTask](#) object (when merging revenue arrangements) or [recognition.MergeElementsTask](#) object (when merging revenue elements). After you obtain one of these objects, you can set its properties, such as the list of arrangements or elements to merge, the date on the merged revenue arrangement, whether to prospectively merge arrangements, and so on. You can use these properties to specify the same input data that you can specify when you merge revenue arrangements using the NetSuite UI. After you set its properties, you can submit the task for processing. Merge tasks are processed asynchronously.

You can use the [recognition.checkStatus\(options\)](#) method to check the status of a submitted merge task. This method returns a [recognition.MergeArrangementsTaskStatus](#) object that describes the current status of the merge task (pending, processing, complete, or failed). This object represents the current status for either a [recognition.MergeArrangementsTask](#) or a [recognition.MergeElementsTask](#). If the task completes successfully, this object includes the ID of the merged revenue arrangement record that was created. If the task fails, this object includes an error message that describes the failure.

To merge revenue arrangements or revenue elements using the N/task/accounting/recognition module, the following requirements must be met:

- The Advanced Revenue Management feature must be enabled in your account. For more information, see the help topic [Enabling the Advanced Revenue Management Feature](#).
- Your role must have the (Transactions) Revenue Arrangement permission assigned at a level of Create or higher. For more information, see the help topic [NetSuite Permissions Overview](#).

For more information about revenue arrangements, see the following help topics:

- [Revenue Arrangement Management](#): This topic describes revenue arrangements in general.
- [Combination and Modification of Performance Obligations](#): This topic describes the different types of merge results (combined revenue arrangements and prospective change orders).
- [Revenue Arrangement](#): This topic describes the revenue arrangement record type, including scripting considerations, supported script types, and sublist fields.

#### In this help topic

- [N/task/accounting/recognition Module Members](#)
- [MergeArrangementsTask Object Members](#)
- [MergeArrangementsTaskStatus Object Members](#)
- [MergeElementsTask Object Members](#)
- [N/task/accounting/recognition Module Script Samples](#)
  - [Sample 1: Merge revenue elements using internal IDs](#)
  - [Sample 2: Merge revenue arrangements using a saved search](#)
  - [Sample 3: Merge revenue arrangements using an ad-hoc search](#)

## N/task/accounting/recognition Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">recognition.MergeArrangementsTask</a>	Object	Server-side scripts	Encapsulates a task to merge all of the revenue elements from a specified list of revenue arrangements.  Use <a href="#">recognition.create(options)</a> to create this object.
	<a href="#">recognition.MergeArrangementsTaskStatus</a>	Object	Server-side scripts	Encapsulates the current status of a submitted merge task.  Use <a href="#">recognition.checkStatus(options)</a> to create this object.
	<a href="#">recognition.MergeElementsTask</a>	Object	Server-side scripts	Encapsulates a task to merge all of the specified revenue elements.  Use <a href="#">recognition.create(options)</a> to create this object.
Method	<a href="#">recognition.create(options)</a>	<a href="#">recognition.MergeArrangementsTask</a>   <a href="#">recognition.MergeElementsTask</a>	Server-side scripts	Creates a merge task that combines entire revenue arrangements or individual revenue elements.  Use values in the <a href="#">recognition.TaskType</a> enum to specify the type of merge task to create.
	<a href="#">recognition.checkStatus(options)</a>	<a href="#">recognition.MergeArrangementsTaskStatus</a>	Server-side scripts	Checks the status of a submitted merge task.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Enum	recognition.TaskStatus	enum	Server-side scripts	Holds the string values for supported merge task statuses.  This enum is used to represent the task status in a <code>recognition.MergeArrangementsTaskStatus</code> object.
	recognition.TaskType	enum	Server-side scripts	Holds the string values for supported merge task types.  This enum is used to pass the task type argument to <code>recognition.create(options)</code> .

## MergeArrangementsTask Object Members

The following members are called on the `recognition.MergeArrangementsTask` object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<code>MergeArrangementsTask.submit()</code>	number (read-only)	Server-side scripts	Submits the merge task for processing.  This method returns a task ID that uniquely identifies the merge task.
Property	<code>MergeArrangementsTask.arrangements</code>	Array<number   string> (read-only)	Server-side scripts	Holds an array of internal IDs of the revenue arrangement records to merge.
	<code>MergeArrangementsTask.contractAcquisitionExpenseAccount</code>	number   string (read-only)	Server-side scripts	References the contract acquisition expense account for the new revenue arrangement. This property is valid only if the accounting preference Enable Advanced Cost Amortization is enabled. For more information, see the help topic <a href="#">Advanced Cost Amortization</a> .  The default value is the account specified by the accounting preference Contract Acquisition Expense Account in your account.
	<code>MergeArrangementsTask.contractAcquisitionDeferredExpenseAccount</code>	number   string (read-only)	Server-side scripts	References the contract acquisition deferred expense account for the new revenue arrangement. This property is valid only if the accounting preference Enable Advanced Cost Amortization is enabled. For more information, see the help topic <a href="#">Advanced Cost Amortization</a> .  The default value is the account specified by the accounting preference Contract Acquisition Deferred Expense Account in your account.
	<code>MergeArrangementsTask.contractCostAccrualDate</code>	JavaScript Date (read-only)	Server-side scripts	Describes the contract cost accrual date to use for the new revenue arrangement. This property is valid only if the accounting preference Enable Advanced Cost Amortization is enabled. For more information, see the help topic <a href="#">Advanced Cost Amortization</a> .  The default value is today's date.
	<code>MergeArrangementsTask.mergeResidualRevenueAmounts</code>	boolean (read-only)	Server-side scripts	Indicates whether the revenue arrangements are merged prospectively. For more information about prospective merges, see the help topic <a href="#">Prospective Merges</a> .  The default value is <code>false</code> .
	<code>MergeArrangementsTask.recalculateResidualFairValue</code>	boolean (read-only)	Server-side scripts	Indicates whether to recalculate the fair value on residual elements when revenue arrangements are prospectively merged. For more information

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
				<p>about prospective merges, see the help topic <a href="#">Prospective Merges</a>.</p> <p>This property can be set to <code>true</code> only if the <a href="#">MergeArrangementsTask.mergeResidualRevenueAmounts</a> property is also set to <code>true</code>. The default value is <code>false</code>.</p>
	<a href="#">MergeArrangementsTask.revenueArrangementDate</a>	JavaScript Date (read-only)	Server-side scripts	<p>Describes the date of the new revenue arrangement.</p> <p>The default value is today's date.</p>

## MergeArrangementsTaskStatus Object Members

The following members are called on the [recognition.MergeArrangementsTaskStatus](#) object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	<a href="#">MergeArrangementsTaskStatus.errorMessage</a>	string (read-only)	Server-side scripts	Holds an error message that describes the failure of the merge task. This property is valid only if the value of the status property is <a href="#">TaskStatus.FAILED</a> .
	<a href="#">MergeArrangementsTaskStatus.inputArrangements</a>	number[] (read-only)	Server-side scripts	Holds an array of internal IDs of the revenue arrangement records to merge. This property is valid only if the merge task was created using a task type of <a href="#">TaskType.MERGE_ARRANGEMENTS_TASK</a> .
	<a href="#">MergeArrangementsTaskStatus.inputElements</a>	number[] (read-only)	Server-side scripts	Holds an array of internal IDs of the revenue elements to merge. This property is valid only if the merge task was created using a task type of <a href="#">TaskType.MERGE_ELEMENTS_TASK</a> .
	<a href="#">MergeArrangementsTaskStatus.resultingArrangement</a>	number   string (read-only)	Server-side scripts	References the internal ID of the new revenue arrangement that was created. This property is valid only if the value of the status property is <a href="#">TaskStatus.COMPLETE</a> .
	<a href="#">MergeArrangementsTaskStatus.status</a>	string (read-only)	Server-side scripts	Represents the current status of the merge task. This property uses values in the <a href="#">recognition.TaskType</a> num.
	<a href="#">MergeArrangementsTaskStatus.submissionId</a>	number   string (read-only)	Server-side scripts	References the submission ID of the merge arrangements bulk process.
	<a href="#">MergeArrangementsTaskStatus.taskId</a>	number   string (read-only)	Server-side scripts	Holds the task ID of the merge task. The task ID is assigned to the merge task when you call <a href="#">recognition.create(options)</a> .

## MergeElementsTask Object Members

The following members are called on the [recognition.MergeElementsTask](#) object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	MergeElementsTask.submit()	number (read-only)	Server-side scripts	Submits the merge task for processing.  This method returns a task ID that uniquely identifies the merge task.
Property	MergeElementsTask.contractAcquisitionExpenseAccount	number   string (read-only)	Server-side scripts	References the contract acquisition expense account for the new revenue arrangement. This property is valid only if the accounting preference Enable Advanced Cost Amortization is enabled. For more information, see the help topic <a href="#">Advanced Cost Amortization</a> .  The default value is the account specified by the accounting preference Contract Acquisition Expense Account in your account.
	MergeElementsTask.contractAcquisitionDeferredExpenseAccount	number   string (read-only)	Server-side scripts	References the contract acquisition deferred expense account for the new revenue arrangement. This property is valid only if the accounting preference Enable Advanced Cost Amortization is enabled. For more information, see the help topic <a href="#">Advanced Cost Amortization</a> .  The default value is the account specified by the accounting preference Contract Acquisition Deferred Expense Account in your account.
	MergeElementsTask.contractCostAccrualDate	JavaScript Date (read-only)	Server-side scripts	Describes the contract cost accrual date to use for the new revenue arrangement. This property is valid only if the accounting preference Enable Advanced Cost Amortization is enabled. For more information, see the help topic <a href="#">Advanced Cost Amortization</a> .  The default value is today's date.
	MergeElementsTask.elements	Array<number   string> (read-only)	Server-side scripts	Holds an array of internal IDs of the revenue element records to merge.
	MergeElementsTask.revenueArrangementDate	JavaScript Date (read-only)	Server-side scripts	Describes the date of the new revenue arrangement.  The default value is today's date.

## N/task/accounting/recognition Module Script Samples

The following script samples demonstrate how to use the features of the N/task/accounting/recognition module.

### Sample 1: Merge revenue elements using internal IDs



**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample adds the internal IDs of several revenue element records to an array. It calls `recognition.create(options)` to create a merge task for revenue element records, uses the array as the list

of revenue element records to merge, and submits the merge task. The sample also checks the status of the merge task.

If you run this sample code in your account, make sure to use the internal IDs of valid revenue element records in your account.

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

require(['N/task/accounting/recognition'], function(recognition){
    var elementsList = [];
    elementsList.push(401);
    elementsList.push(402);

    var recognitionTask = recognition.create({
        taskType: recognition.TaskType.MERGE_ELEMENTS_TASK
    });
    recognitionTask.elements = elementsList;
    var taskStatusId = recognitionTask.submit();

    var mergeTaskState = recognition.checkStatus({
        taskId: taskStatusId
    });

    log.debug('Submission ID = ' + mergeTaskState.submissionId);
    log.debug('Resulting Arrangement ID = ' + mergeTaskState.resultingArrangement);
    log.debug('status = ' + mergeTaskState.status);
});
});
```

## Sample 2: Merge revenue arrangements using a saved search

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample loads a saved search for revenue arrangement records. It obtains the value of the `internalid` field from each record in the result set, and it adds the values to an array. It calls `recognition.create(options)` to create a merge task for revenue arrangement records, uses the array as the list of revenue arrangement records to merge, and submits the merge task. The sample also checks the status of the merge task and logs status information.

If you run this sample code in your account, make sure to use a saved search for valid revenue arrangement records in your account.

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

require(['N/task/accounting/recognition', 'N/search'], function(recognition, search){
    var mySearch = search.load({
        id: 'customsearch22'
    });
});
```

```

var elementsList = [];
mySearch.run().each(function(result) {
    var id = result.getValue({
        name: 'internalid'
    });
    elementsList.push(id);
});

var recognitionTask = recognition.create({
    taskType: recognition.TaskType.MERGE_ARRANGEMENTS_TASK
});

recognitionTask.arrangements = elementsList;
recognitionTask.revenueArrangementDate = new Date(2019, 2, 10);

var taskStatusId = recognitionTask.submit();
log.debug('taskId = ' + taskStatusId);

var mergeTaskState = recognition.checkStatus({
    taskId: taskStatusId
});

log.debug('Submission ID = ' + mergeTaskState.submissionId);
log.debug('Resulting Arrangement ID = ' + mergeTaskState.resultingArrangement);
log.debug('status = ' + mergeTaskState.status);
log.debug('Error message = ' + mergeTaskState.errorMessage);
});

```

## Sample 3: Merge revenue arrangements using an ad-hoc search

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample creates an ad-hoc search for revenue element records. It obtains the first 50 results, obtains the value of the `elementsList` field from each record in the result set, and adds the values to an array. It calls `recognition.create(options)` to create a merge task for revenue element records, uses the array as the list of revenue elements records to merge, and submits the merge task. This sample also checks the status of the merge task and logs status information.

```

/**
 * @NApiVersion 2.x
 */

require(['N/task/accounting/recognition', 'N/search'], function(recognition, search) {
    var elementsList = [];
    var rs = search.create({
        type: 'revenueelement',
        columns: [
            'internalid'
        ]
    }).run();

```

```

var results = rs.getRange(0, 50);
for (var i = 0; i < results.length; i++) {
    var id = result.getValue('elementsList');
    elementsList.push(id);
}

var t = recognition.create({
    taskType: recognition.TaskType.MERGE_ELEMENTS_TASK
});
t.elements = elementsList;
t.revenueArrangementDate = new Date(2019, 1, 1);

var taskId = t.submit();
log.debug('Initial status: ' + res.status);
});

```

## recognition.MergeArrangementsTask

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a task to merge all of the revenue elements from a specified list of revenue arrangements.  Use <a href="#">recognition.create(options)</a> to create this object. After you create the object, you can populate its properties and submit the task for processing. The <a href="#">MergeArrangementsTask.arrangements</a> property is required, and all other properties are optional.
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>
<b>Methods and Properties</b>	<a href="#">MergeArrangementsTask Object Members</a>
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## MergeArrangementsTask.submit()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Submits the merge task for processing.
---------------------------	--

	<p>This method returns a task ID that uniquely identifies the merge task. This task ID also represents the submission ID of the internal bulk process that performs the merge.</p> <p>Before you call this method to submit a merge task, you must populate the properties of the <code>recognition.MergeArrangementsTask</code> object (such as <code>MergeArrangementsTask.arrangements</code>, <code>MergeArrangementsTask.contractAcquisitionExpenseAccount</code>, and so on).</p>
<b>Returns</b>	number
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	20 usage units
<b>Module</b>	N/task/accounting/recognition Module
<b>Parent Object</b>	<code>recognition.MergeArrangementsTask</code>
<b>Sibling Object Members</b>	<a href="#">MergeArrangementsTask Object Members</a>
<b>Since</b>	2019.2

## Errors

Error Code	Thrown If
NO_REVENUE_ARRANGEMENT_IDS_ARE_INCLUDED_IN_YOUR_INPUT	The <code>MergeArrangementsTask.arrangements</code> property is empty.
NO_REVENUE_ELEMENTS_WERE_FOUND_FOR_THE_REVENUER_ARRANGEMENT_IDS_YOU_INPUT	No revenue elements were found for the revenue arrangements specified in the <code>MergeArrangementsTask.arrangements</code> property.
WRONG_PARAMETER_TYPE	An invalid date format is specified in the <code>MergeArrangementsTask.contractCostAccrualDate</code> property or the <code>MergeArrangementsTask.revenueArrangementDate</code> property.

## Syntax



**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](#).

TBD

## MergeArrangementsTask.arrangements



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>Holds an array of internal IDs of the revenue arrangement records to merge.</p> <p>This property is required. You must specify a value for this property before you can submit the task for processing using <code>MergeArrangementsTask.submit()</code>.</p> <p>If you do not specify any revenue arrangement record IDs, a <code>NO_REVENUE_ARRANGEMENT_IDS_ARE_INCLUDED_IN_YOUR_INPUT</code> error is thrown when you call <code>MergeArrangementsTask.submit()</code>.</p>
-----------------------------	---

	for the task. Invalid IDs are ignored. If no revenue elements were found for the specified revenue arrangement record IDs, a <b>NO_REVENUE_ELEMENTS_WERE_FOUND_FOR_THE_REVENUERARRANGEMENT_IDS_YOU_INPUT</b> error is thrown when you call <a href="#">MergeArrangementsTask.submit()</a> .
<b>Type</b>	Array<number   string>
<b>Module</b>	N/task/accounting/recognition Module
<b>Parent Object</b>	recognition.MergeArrangementsTask
<b>Sibling Object Members</b>	<a href="#">MergeArrangementsTask Object Members</a>
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## MergeArrangementsTask.contractAcquisitionExpenseAccount



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	References the contract acquisition expense account for the new revenue arrangement. This property is valid only if the accounting preference Enable Advanced Cost Amortization is enabled. For more information, see the help topic <a href="#">Advanced Cost Amortization</a> . If this preference is not enabled, this property is ignored.  This property is optional. The default value is the account specified by the accounting preference Contract Acquisition Expense Account in your account.
<b>Type</b>	number   string
<b>Module</b>	N/task/accounting/recognition Module
<b>Parent Object</b>	recognition.MergeArrangementsTask
<b>Sibling Object Members</b>	<a href="#">MergeArrangementsTask Object Members</a>
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## MergeArrangementsTask.contractAcquisitionDeferredExpenseAccount

<p><b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.</p>	
<b>Property Description</b>	References the contract acquisition deferred expense account for the new revenue arrangement. This property is valid only if the accounting preference Enable Advanced Cost Amortization is enabled. For more information, see the help topic <a href="#">Advanced Cost Amortization</a> . If this preference is not enabled, this property is ignored.  This property is optional. The default value is the account specified by the accounting preference Contract Acquisition Deferred Expense Account in your account.
<b>Type</b>	number   string
<b>Module</b>	N/task/accounting/recognition Module
<b>Parent Object</b>	recognition.MergeArrangementsTask
<b>Sibling Object Members</b>	<a href="#">MergeArrangementsTask Object Members</a>
<b>Since</b>	2019.2

### 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/accounting/recognition Module Script Samples](#).

TBD

## MergeArrangementsTask.contractCostAccrualDate

<p><b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.</p>	
<b>Property Description</b>	Describes the contract cost accrual date to use for the new revenue arrangement. This property is valid only if the accounting preference Enable Advanced Cost Amortization is enabled. For more information, see the help topic <a href="#">Advanced Cost Amortization</a> . If this preference is not enabled, this property is ignored.  This property is optional. The default value is today's date. If you specify an invalid date format, a <code>WRONG_PARAMETER_TYPE</code> error is thrown when you call <a href="#">MergeArrangementsTask.submit()</a> for the task.
<b>Type</b>	JavaScript Date
<b>Module</b>	N/task/accounting/recognition Module
<b>Parent Object</b>	recognition.MergeArrangementsTask
<b>Sibling Object Members</b>	<a href="#">MergeArrangementsTask Object Members</a>
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## MergeArrangementsTask.mergeResidualRevenueAmounts



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether the revenue arrangements are merged prospectively. For more information about prospective merges, see the help topic <a href="#">Prospective Merges</a> .  This property is optional. The default value is <code>false</code> .
<b>Type</b>	boolean
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>
<b>Parent Object</b>	<a href="#">recognition.MergeArrangementsTask</a>
<b>Sibling Object Members</b>	<a href="#">MergeArrangementsTask Object Members</a>
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## MergeArrangementsTask.recalculateResidualFairValue



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether to recalculate the fair value on residual elements when revenue arrangements are prospectively merged. For more information about prospective merges, see the help topic <a href="#">Prospective Merges</a> .  This property is optional. This property can be set to <code>true</code> only if the <a href="#">MergeArrangementsTask.mergeResidualRevenueAmounts</a> property is also set to <code>true</code> . If the <a href="#">MergeArrangementsTask.mergeResidualRevenueAmounts</a> property is <code>false</code> , this property is ignored. The default value of this property is <code>false</code> .
<b>Type</b>	boolean
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>
<b>Parent Object</b>	<a href="#">recognition.MergeArrangementsTask</a>

<b>Sibling Object Members</b>	MergeArrangementsTask Object Members
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## MergeArrangementsTask.revenueArrangementDate



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Describes the date of the new revenue arrangement.  This property is optional. The default value is today's date. If you specify an invalid date format, a <code>WRONG_PARAMETER_TYPE</code> error is thrown when you call <code>MergeArrangementsTask.submit()</code> for the task.
<b>Type</b>	JavaScript Date
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>
<b>Parent Object</b>	<a href="#">recognition.MergeArrangementsTask</a>
<b>Sibling Object Members</b>	<a href="#">MergeArrangementsTask Object Members</a>
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## recognition.MergeArrangementsTaskStatus



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the current status of a submitted merge task.  Use <code>recognition.checkStatus(options)</code> to create this object. The current status corresponds to one of the values in the <code>recognition.TaskStatus</code> enum: <code>PENDING</code> , <code>PROCESSING</code> , <code>COMPLETE</code> , or <code>FAILED</code> .
---------------------------	---

<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>
<b>Methods and Properties</b>	<a href="#">MergeArrangementsTaskStatus Object Members</a>
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## MergeArrangementsTaskStatus.errorMessage



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Holds an error message that describes the failure of the merge task. This property is valid only if the value of the <a href="#">MergeArrangementsTaskStatus.status</a> property is <a href="#">TaskStatus.FAILED</a> .
<b>Type</b>	string (read-only)
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>
<b>Parent Object</b>	<a href="#">recognition.MergeArrangementsTaskStatus</a>
<b>Sibling Object Members</b>	<a href="#">MergeArrangementsTaskStatus Object Members</a>
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## MergeArrangementsTaskStatus.inputArrangements



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Holds an array of internal IDs of the revenue arrangement records to merge. This property is valid only if the merge task was created using a task type of <a href="#">TaskType.MERGE_ARRANGEMENTS_TASK</a> .
-----------------------------	---

	 <b>Note:</b> This property has a potential governance value of 10 usage units. For more information about governance, see the help topic <a href="#">SuiteScript Governance</a> .
<b>Type</b>	number[] (read-only)
<b>Module</b>	N/task/accounting/recognition Module
<b>Parent Object</b>	recognition.MergeArrangementsTaskStatus
<b>Sibling Object Members</b>	<a href="#">MergeArrangementsTaskStatus Object Members</a>
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## MergeArrangementsTaskStatus.inputElements



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Holds an array of internal IDs of the revenue elements to merge. This property is valid only if the merge task was created using a task type of <code>TaskType.MERGE_ELEMENTS_TASK</code> .
<b>Type</b>	number[] (read-only)
<b>Module</b>	N/task/accounting/recognition Module
<b>Parent Object</b>	recognition.MergeArrangementsTaskStatus
<b>Sibling Object Members</b>	<a href="#">MergeArrangementsTaskStatus Object Members</a>
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## MergeArrangementsTaskStatus.resultingArrangement

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	References the internal ID of the new revenue arrangement that was created. This property is valid only if the value of the <a href="#">MergeArrangementsTaskStatus.status</a> property is <code>TaskStatus.COMPLETED</code> .
<b>Type</b>	number (read-only)
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>
<b>Parent Object</b>	<a href="#">recognition.MergeArrangementsTaskStatus</a>
<b>Sibling Object Members</b>	<a href="#">MergeArrangementsTaskStatus Object Members</a>
<b>Since</b>	2019.2

### 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/accounting/recognition Module Script Samples](#).

TBD

## MergeArrangementsTaskStatus.status

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Represents the current status of the merge task. This property uses values in the <a href="#">recognition.TaskStatus</a> enum.
<b>Type</b>	string (read-only)
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>
<b>Parent Object</b>	<a href="#">recognition.MergeArrangementsTaskStatus</a>
<b>Sibling Object Members</b>	<a href="#">MergeArrangementsTaskStatus Object Members</a>
<b>Since</b>	2019.2

### 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/accounting/recognition Module Script Samples](#).

TBD

## MergeArrangementsTaskStatus.submissionId



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	References the submission ID of the merge arrangements bulk process.  This ID is the same as the task ID that is returned by <a href="#">MergeArrangementsTask.submit()</a> or <a href="#">MergeElementsTask.submit()</a> .
<b>Type</b>	number (read-only)
<b>Module</b>	N/task/accounting/recognition Module
<b>Parent Object</b>	recognition.MergeArrangementsTaskStatus
<b>Sibling Object Members</b>	<a href="#">MergeArrangementsTaskStatus Object Members</a>
<b>Since</b>	2019.2

### 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/accounting/recognition Module Script Samples](#).

TBD

## MergeArrangementsTaskStatus.taskId



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Holds the task ID of the merge task. The task ID is assigned to the merge task when you call <a href="#">MergeArrangementsTask.submit()</a> or <a href="#">MergeElementsTask.submit()</a> for the task.
<b>Type</b>	number   string (read-only)
<b>Module</b>	N/task/accounting/recognition Module
<b>Parent Object</b>	recognition.MergeArrangementsTaskStatus
<b>Sibling Object Members</b>	<a href="#">MergeArrangementsTaskStatus Object Members</a>
<b>Since</b>	2019.2

### 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/accounting/recognition Module Script Samples](#).

TBD

# recognition.MergeElementsTask



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a task to merge all of the specified revenue elements. Use <a href="#">recognition.create(options)</a> to create this object. After you create the object, you can populate its properties and submit the task for processing. The <a href="#">MergeElementsTask.elements</a> property is required, and all other properties are optional.
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>
<b>Methods and Properties</b>	<a href="#">MergeElementsTask Object Members</a>
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## MergeElementsTask.submit()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Submits the merge task for processing.  This method returns a task ID that uniquely identifies the merge task. This task ID also represents the submission ID of the internal bulk process that performs the merge.
<b>Returns</b>	number
<b>Supported Script Types</b>	Server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	20 usage units
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>
<b>Parent Object</b>	<a href="#">recognition.MergeElementsTask</a>
<b>Sibling Object Members</b>	<a href="#">MergeElementsTask Object Members</a>
<b>Since</b>	2019.2

## Errors

Error Code	Thrown If
WRONG_PARAMETER_TYPE	An invalid date format is specified in the <a href="#">MergeElementsTask.contractCostAccrualDate</a> property or the <a href="#">MergeElementsTask.revenueArrangementDate</a> property.

## Syntax



**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](#).

TBD

## MergeElementsTask.contractAcquisitionExpenseAccount



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	References the contract acquisition expense account for the new revenue arrangement. This property is valid only if the accounting preference Enable Advanced Cost Amortization is enabled. For more information, see the help topic <a href="#">Advanced Cost Amortization</a> . If this preference is not enabled, this property is ignored.  This property is optional. The default value is the account specified by the accounting preference Contract Acquisition Expense Account in your account.
<b>Type</b>	number   string
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>
<b>Parent Object</b>	<a href="#">recognition.MergeElementsTask</a>
<b>Sibling Object Members</b>	<a href="#">MergeElementsTask Object Members</a>
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## MergeElementsTask.contractAcquisitionDeferredExpenseAccount



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	References the contract acquisition deferred expense account for the new revenue arrangement. This property is valid only if the accounting preference Enable Advanced Cost
-----------------------------	---

	<p>Amortization is enabled. For more information, see the help topic <a href="#">Advanced Cost Amortization</a>. If this preference is not enabled, this property is ignored.</p> <p>This property is optional. The default value is the account specified by the accounting preference Contract Acquisition Deferred Expense Account in your account.</p>
<b>Type</b>	number   string
<b>Module</b>	N/task/accounting/recognition Module
<b>Parent Object</b>	recognition.MergeElementsTask
<b>Sibling Object Members</b>	MergeElementsTask Object Members
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## MergeElementsTask.contractCostAccrualDate



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Describes the contract cost accrual date to use for the new revenue arrangement. This property is valid only if the accounting preference Enable Advanced Cost Amortization is enabled. For more information, see the help topic <a href="#">Advanced Cost Amortization</a> .  This property is optional. The default value is today's date. If you specify an invalid date format, a <code>WRONG_PARAMETER_TYPE</code> error is thrown when you call <code>MergeElementsTask.submit()</code> for the task.
<b>Type</b>	JavaScript Date
<b>Module</b>	N/task/accounting/recognition Module
<b>Parent Object</b>	recognition.MergeElementsTask
<b>Sibling Object Members</b>	MergeElementsTask Object Members
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## MergeElementsTask.elements



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Holds an array of internal IDs of the revenue element records to merge.  This property is required. You must specify a value for this property before you can submit the task for processing using <a href="#">MergeElementsTask.submit()</a> .
<b>Type</b>	Array<number   string>
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>
<b>Parent Object</b>	<a href="#">recognition.MergeElementsTask</a>
<b>Sibling Object Members</b>	<a href="#">MergeElementsTask Object Members</a>
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## MergeElementsTask.revenueArrangementDate



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Describes the date of the new revenue arrangement.  This property is optional. The default value is today's date. If you specify an invalid date format, a <code>WRONG_PARAMETER_TYPE</code> error is thrown when you call <a href="#">MergeElementsTask.submit()</a> for the task.
<b>Type</b>	JavaScript Date
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>
<b>Parent Object</b>	<a href="#">recognition.MergeElementsTask</a>
<b>Sibling Object Members</b>	<a href="#">MergeElementsTask Object Members</a>
<b>Since</b>	2019.2

## 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/accounting/recognition Module Script Samples](#).

TBD

## recognition.create(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a merge task that combines entire revenue arrangements or individual revenue elements.  Use values in the <a href="#">recognition.TaskType</a> enum to specify the type of merge task to create. After you call this method to create a merge task, you must specify the properties of the merge task (such as <a href="#">MergeArrangementsTask.arrangements</a> or <a href="#">MergeElementsTask.elements</a> ) before you submit the task using <a href="#">MergeArrangementsTask.submit()</a> or <a href="#">MergeElementsTask.submit()</a> .
<b>Returns</b>	<a href="#">recognition.MergeArrangementsTask</a>   <a href="#">recognition.MergeElementsTask</a>
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>
<b>Sibling Module Members</b>	<a href="#">N/task/accounting/recognition Module Members</a>
<b>Since</b>	2019.2

## Parameters



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

Parameter	Type	Required / Optional	Description
options.taskType	string	required	The type of merge task to create.  Use values from the <a href="#">recognition.TaskType</a> enum for this parameter.

## Errors

Error Code	Thrown If
INVALID_TASK_TYPE	The <code>options.taskType</code> parameter represents an invalid task 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/task/accounting/recognition Module Script Samples](#).

TBD

## recognition.checkStatus(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Checks the status of a submitted merge task.
<b>Returns</b>	<a href="#">recognition.MergeArrangementsTaskStatus</a>
<b>Supported Script Types</b>	Server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	50 usage units
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>
<b>Sibling Module Members</b>	<a href="#">N/task/accounting/recognition Module Members</a>
<b>Since</b>	2019.2

## Parameters



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

Parameter	Type	Required / Optional	Description
options.taskId	number   string	required	The task ID of the merge task to check.  The task ID is assigned to the merge task when you call <a href="#">recognition.create(options)</a> .

## Syntax



**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/accounting/recognition Module Script Samples](#).

TBD

## recognition.TaskStatus



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the string values for supported merge task statuses.  This enum is used to represent the task status in a <a href="#">recognition.MergeArrangementsTaskStatus</a> object.
-------------------------	---

	<b>Note:</b> 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
Module	<a href="#">N/task/accounting/recognition Module</a>
Sibling Module Members	<a href="#">N/task/accounting/recognition Module Members</a>
Since	2019.2

## Values

Value
COMPLETE
FAILED
PENDING
PROCESSING

## Syntax



**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/accounting/recognition Module Script Samples](#).

TBD

## recognition.TaskType



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Holds the string values for supported merge task types.  This enum is used to pass the task type argument to <a href="#">recognition.create(options)</a> .
<b>Type</b>	enum
<b>Module</b>	<a href="#">N/task/accounting/recognition Module</a>

<b>Sibling Module Members</b>	N/task/accounting/recognition Module Members
<b>Since</b>	2019.2

## Values

Value
MERGE_ARRANGEMENTS_TASK
MERGE_ELEMENTS_TASK

## Syntax

**⚠ 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/accounting/recognition Module Script Samples](#).

TBD

## N/transaction Module

**ℹ Note:** The content in this help topic pertains to SuiteScript 2.0.

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 Samples](#)

### N/transaction Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<code>transaction.void(options)</code>	number	Client and server-side scripts	Voids a transaction record.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	transaction_void.promise(options)	number	Client scripts	Voids a transaction record asynchronously.
Enum	transaction.Type	enum	Client and server-side scripts	Enumeration that holds the string values for supported record types.

## N/transaction Module Script Samples

The following script samples demonstrate how to use the features of the N/transaction module.

### Sample 1: Void a transaction

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample creates a sales order record, saves it, then voids the sales order. Before creating the sales order record, the sample loads a set of accounting preferences from the current NetSuite account, specifies that the **REVERSALVOIDING** preference should be disabled (set to `false`), and saves the preferences. This sample works only in NetSuite OneWorld accounts. Make sure to replace hard-coded values (such as record IDs) with valid values from your NetSuite account.

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

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: 107
        });
    }
});
```

```

        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
    });

    // The value of the memo field is 'VOID'
    var memo = salesOrder.getValue({
        fieldId: 'memo'
    });
}

voidSalesOrder();
});

```

## transaction.void(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Method used to void a transaction record object and return an id that indicates the type of void performed.</p> <p>The type of void performed depends on the targeted account's preference settings.</p> <div style="border: 1px solid #f0e68c; padding: 5px; margin-top: 10px;">  <b>Important:</b> After you void a transaction, you cannot make changes to the transaction that impact the general ledger.       </div>
<b>Returns</b>	<p>An ID returned as a number.</p> <ul style="list-style-type: none"> <li>■ If a direct void is performed, returns the ID of the record voided.</li> <li>■ If a void by reversing journal is performed, returns the ID of the newly created voiding journal.</li> </ul>
<b>Supported Script Types</b>	<p>All client and server-side scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Governance</b>	10 units

<b>Module</b>	N/transaction Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.id	number   string	required	Internal ID of the specific transaction record instance to void.	2015.2
options.type	string	required	Internal ID of the type of transaction record to void	2015.2

## Errors

Error Code	Message	Thrown If
INVALID_RECORD_TYPE		The <b>type</b> argument passed is not valid or the record type is not voidable.
THAT_RECORD_DOES_NOT_EXIST		The <b>id</b> argument passed is not valid.
SSS_MISSING_REQD_ARGUMENT		The <b>type</b> or <b>id</b> argument 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 [transaction Module Script Sample](#).

```
//Add additional code
...
var voidSalesOrderId = transaction void({
    type: transaction.Type.SALES_ORDER,
    id: salesOrderId
});
...
//Add additional code
```

## transaction void.promise(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Method used to void a transaction record object asynchronously and return an ID that indicates the type of void performed: <ul style="list-style-type: none"> <li>■ If a direct void is performed, this method returns the ID of the record that was voided.</li> <li>■ If a void by reversing journal is performed, this method returns the ID of the newly created voiding journal.</li> </ul>
---------------------------	--

	<p>The type of void performed depends on the targeted account's preference settings.</p> <p><b>Important:</b> After you void a transaction, you cannot make changes to the transaction that impact the general ledger.</p> <p><b>Note:</b> The parameters and errors thrown for this method are the same as those for <a href="#">transaction.void(options)</a>. For more information on promises, see <a href="#">Promise Object</a>.</p>
<b>Returns</b>	<a href="#">Promise Object</a>
<b>Synchronous Version</b>	<a href="#">transaction.void(options)</a>
<b>Supported Script Types</b>	<p>All client-side scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a>.</p>
<b>Governance</b>	10 units
<b>Module</b>	<a href="#">N/transaction Module</a>
<b>Since</b>	2015.2

## Syntax

**Important:** The following code sample shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```
//Add additional code
...
var voidSalesOrderId = transaction.void.promise({
    type: record.Type.SALES_ORDER,
    id: salesOrderId
});
...
//Add additional code
```

## transaction.Type

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for supported transaction record types.
	<p><b>Note:</b> 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>
<b>Supported Script Types</b>	<p>All client and server-side scripts</p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Module</b>	<a href="#">N/transaction Module</a>

<b>Since</b>	2015.2
--------------	--------

## Values

<b>Transaction Record</b>	<b>Supported Void Type</b>
ASSEMBLY_BUILD	None
ASSEMBLY_UNBUILD	None
BIN_TRANSFER	None
BIN_WORKSHEET	None
BLANKET_PURCHASE_ORDER	None
CASH_REFUND	Direct Void
CASH_SALE	Direct Void
CHECK	Void by Reversing Journal
CREDIT_CARD_CHARGE	None
CREDIT_CARD_REFUND	None
CREDIT_MEMO	Direct Void
CUSTOMER_DEPOSIT	Direct Void
CUSTOMER_PAYMENT	Direct Void
CUSTOMER_PAYMENT_AUTHORIZATION	None
CUSTOMER_REFUND	Direct Void and Void by Reversing Journal
CUSTOM_TRANSACTION	None
DEPOSIT	None
DEPOSIT_APPLICATION	None
ESTIMATE	Direct Void
EXPENSE_REPORT	Direct Void
FULFILLMENT_REQUEST	None
INBOUND_SHIPMENT	None
INVENTORY_ADJUSTMENT	None
INVENTORY_COST_REVALUATION	None
INVENTORY_COUNT	None
INVENTORY_STATUS_CHANGE	None
INVENTORY_TRANSFER	None
INVOICE	Direct Void
ITEM_FULFILLMENT	None

Transaction Record	Supported Void Type
ITEM_RECEIPT	None
JOURNAL_ENTRY	Direct Void
OPPORTUNITY	None
PAYCHECK	None
PAYCHECK_JOURNAL	Direct Void
PERIOD_END_JOURNAL	None
PURCHASE_CONTRACT	None
PURCHASE_ORDER	None
PURCHASE_REQUSITION	None
RETURN_AUTHORIZATION	Direct Void
REVENUE_ARRANGEMENT	None
REVENUE_COMMITMENT	None
REVENUE_COMMITMENT_REVERSAL	None
SALES_ORDER	Direct Void
STORE_PICKUP_FULFILLMENT	None
TRANSFER_ORDER	Direct Void
VENDOR_BILL	Direct Void
VENDOR_CREDIT	Direct Void
VENDOR_PAYMENT	Direct Void and Void by Reversing Journal
VENDOR_RETURN_AUTHORIZATION	Direct Void
WORK_ORDER	Direct Void
WORK_ORDER_CLOSE	Direct Void
WORK_ORDER_COMPLETION	Direct Void
WORK_ORDER_ISSUE	Direct Void

## Syntax



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

```
//Add additional code
...
var voidSalesOrderId = transaction.void({
    type: transaction.Type.SALES_ORDER,
    id: salesOrderId
});
```

```
//Add additional code
```

## N/translation Module



**Note:** The content in this help topic pertains to SuiteScript 2.0.

The N/translation module lets SuiteScript developers interact with NetSuite Translation Collections programmatically. For more information about Translation Collections, see the help topic [Translation Collection Overview](#).

You can watch a video that demonstrates how to use the N/translation module to work with Translation Collections. [View the video](#).

A Translation Collection is a customization object that stores translation strings with their translations. In 2019.1, a single Translation Collection can contain up to 1,000 translation strings. A translation string is a key/value pair where the key is an identifier and its value is a source string. A key references one string that can be translated into multiple languages. For example, a translation string 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 your scripts, as well as in Suitelets and 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 strings stored in Translation Collections. The N/translation module provides read-only access to Translation Collections. 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:

```
// 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 strings 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

Member Type	Name	Return Type / Value Type	Supported Script Type	Description
Object	<code>translation.Handle</code>	Object	Client and server-side scripts	Encapsulates a Translation Collection for a locale.
	<code>translation.Translator</code>	Object / Function	Client and server-side scripts	Represents a translator function that returns translated strings. The translated strings include variables that are passed as parameters to the translator function.
Method	<code>translation.get(options)</code>	<code>translation.Translator</code>	Client and server-side scripts	Creates a translator function for a key in the specified Translation Collection and locale.
	<code>translation.load(options)</code>	<code>translation.Handle</code>	Client and server-side scripts	Creates a <code>translation.Handle</code> object with translations for the specified Translation Collections and locales.
	<code>translation.selectLocale(options)</code>	<code>translation.Handle</code>	Client and server-side scripts	Creates a <code>translation.Handle</code> object in the specified locale from an existing <code>translation.Handle</code> object.
Enum	<code>translation.Locale</code>	enum	Client and server-side scripts	Holds the supported locales for Translation Collections.  This enum is used to pass the locale argument to <code>translation.get(options)</code> and <code>translation.selectLocale(options)</code> .

## N/translation Module Script Samples

See the following script samples for examples of how to use the N/translation module.

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following 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.

```
/*
 * @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
});
});
```

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following 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.

```
/**
 * @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
});
});
```

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following 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 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!"

```
/*
 * @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: 'HELLO_1'
            })(),
            params: ['NetSuite']
        },
        type: message.Type.CONFIRMATION
    });

        // Show the message for 5 seconds
        myMsg.show({
            duration: 5000
        });
});
```

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following 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.

```

/**
 * @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
        });
    });
}

```

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample loads translation strings by key 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.

```

/**
 * @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',
                    keys: ['MY_TITLE']
                },
                {
                    alias: 'myOtherCollection',
                    collection: 'custcollection_other_strings',
                    keys: ['OTHER_MESSAGE']
                }
            ]
        });

        // Create a message with translated strings
        var myMsg = message.create({
            title: localizedStrings.myCollection.MY_TITLE(),
            message: localizedStrings.myOtherCollection.OTHER_MESSAGE(),
            type: message.Type.CONFIRMATION
        });

        // Show the message for 5 seconds
        myMsg.show({
            duration: 5000
        });
    });
}

```

```

        keys: ['MY_OTHER_MESSAGE']
    }]
});

// 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
});
});
});

```

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample loads translation strings by key from 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)`.

```

/**
 * @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',
                keys: ['MY_TITLE', 'MY_MESSAGE']
            },
            {
                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 content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	<p>Encapsulates a Translation Collection for a locale.</p> <p>Use <a href="#">translation.load(options)</a> to create a <b>translation.Handle</b> object with translations for the specified Translation Collections and locales. Use <a href="#">translation.selectLocale(options)</a> to create a <b>translation.Handle</b> object in the specified locale from an existing <b>translation.Handle</b> object.</p> <p>The <b>translation.Handle</b> object is a hierarchical object, which means that each node in the object is either another <b>translation.Handle</b> object or a <a href="#">translation.Translator</a> 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.</p>
<b>Supported Script Types</b>	<p>Client and server-side scripts</p> <p>For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Module</b>	<a href="#">N/translation Module</a>
<b>Sibling Object Members</b>	<a href="#">N/translation Module Members</a>
<b>Since</b>	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/translation Module Script Samples](#).

```

// 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
```

```
/**
 * @NApiVersion 2.0
 * @NScriptType clientscript
 */
define(['N/translation'], function(translation) {
    return {
        pageInit: function(context) {
            var handle = translation.load({
                collections: [
                    {alias: "phrases", collection: "CUSTCOLLECTION_PHRASES",
                     keys: ["HELLO"]},
                    {alias: "specialstrings", collection: "CUSTCOLLECTION_SPECIALSTRINGS",
                     keys: ["HELLO_1"]}
                ],
                locales: ["fr_FR", 'en_US']
            });
            console.log(handle.phrases.HELLO());
            //logs hello in company default language - Hello (if default is English)
            var frenchHandle = translation.selectLocale({handle: result, locale: "fr_FR"});
            console.log(frenchHandle.phrases.HELLO());
            //logs hello in french - Bonjour
        }
    };
}
);
...
// Add additional code
```

## translation.Translator



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object / Function Description</b>	<p>Represents a translator function that returns translated strings.</p> <p>Use <a href="#">translation.get(options)</a> 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.</p> <p>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.</p> <p>For example, “Hello, {1}!” is a valid translation string, where {1} is a placeholder for a parameter. If you call <a href="#">translation.get(options)</a> and specify a parameter of “NetSuite”, the translator function returns “Hello, NetSuite!” in the appropriate locale.</p>
<b>Supported Script Types</b>	Client and server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Module</b>	N/translation Module
<b>Sibling Object Members</b>	N/translation Module Members
<b>Since</b>	2019.1

## Parameters

Parameter	Type	Required / Optional	Description
options.params	string[]	optional	The parameters to pass to the translator function. The parameter values are used in parametrized translation strings.

## Errors

Error Code	Thrown If
WRONG_PARAMETER_TYPE	The function parameters were not passed as an array.

## Syntax

**⚠ 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']
},
    type: message.Type.CONFIRMATION
});
...
// Add additional code
```

## translation.get(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <code>translation.get(options)</code> and do not specify a locale, the method uses the current user's session locale. You can use the <code>options.locale</code> parameter to specify another locale. The <code>translation.Locale</code> enum lists all locales that are enabled for a company, and you can use these locales in <code>translation.get(options)</code> .
Returns	<code>translation.Translator</code>
Supported Script Types	Client and server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
Module	<a href="#">N/translation Module</a>
Sibling Object Members	<a href="#">N/translation Module Members</a>
Since	2019.1

## Parameters

Parameter	Type	Required / Optional	Description
<code>options.collection</code>	string	required	The script ID of the collection.
<code>options.key</code>	string	required	A valid key from the collection.
<code>options.locale</code>	string	optional	A valid locale from the <code>translation.Locale</code> enum.  If a locale is not specified, the locale from the current session is used as the default locale.

## Errors

Error Code	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	A collection or key parameter is missing.
<code>INVALID_TRANSLATION_KEY</code>	The format of a specified key is invalid.
<code>INVALID_TRANSLATION_COLLECTION</code>	The format of a specified collection is invalid.
<code>INVALID_LOCALE</code>	The format of a specified locale is invalid.
<code>TRANSLATION_KEY_NOT_FOUND</code>	A specified translation key was not found.

## Syntax

**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']
    }),
    type: message.Type.CONFIRMATION
});
...
// Add additional code

```

## translation.load(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>Creates a <a href="#">translation.Handle</a> object with translations for the specified Translation Collections and locales.</p> <p>This method returns a <a href="#">translation.Handle</a> object with translation strings organized by collection and ID. Every node in a <a href="#">translation.Handle</a> object is either another <a href="#">translation.Handle</a> object or a <a href="#">translation.Translator</a> function.</p> <p>You can load translation strings from multiple Translation Collections in a single call of <a href="#">translation.load(options)</a>. You must specify the keys of individual translation strings that you want to load. You cannot load all of the terms in a Translation Collection at one time. The <a href="#">translation.load(options)</a> method can load a maximum of 1,000 translation strings, regardless of whether the strings are loaded from one collection or multiple collections.</p> <p>When you load translation strings using <a href="#">translation.load(options)</a>, you can specify a list of valid locales for the strings. You can use these locales when you select a locale using <a href="#">translation.selectLocale(options)</a>. If you specify more than one locale when you call <a href="#">translation.load(options)</a>, the first specified locale in the list is used for the created <a href="#">translation.Handle</a> object. If you want to use a different locale from the list, use <a href="#">translation.selectLocale(options)</a>, which returns a <a href="#">translation.Handle</a> object in the specified locale. You must load a locale using <a href="#">translation.load(options)</a> before you can select it using <a href="#">translation.selectLocale(options)</a>.</p>
<b>Returns</b>	<a href="#">translation.Handle</a>
<b>Supported Script Types</b>	Client and server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/translation Module</a>
<b>Sibling Object Members</b>	<a href="#">N/translation Module Members</a>
<b>Since</b>	2019.1

## Parameters

Parameter	Type	Required / Optional	Description
<code>options.collections</code>	<code>Object[]</code>	required	A list of <a href="#">translation.Handle</a> objects to load.

Parameter	Type	Required / Optional	Description
options.collections.alias	string	required	An alias to identify the collection. This alias is used by the script to determine the collection to load.
options.collections.collection	string	required	The script ID of the collection to load.
options.collections.keys	string[]	required	A list of translation keys from the collection to load.
options.locales	string[]	optional	A list of locales to load the collection in. Use the values in the <a href="#">translation.Locale</a> enum to set this value.

## Errors

Error Code	Thrown If
WRONG_PARAMETER_TYPE	One of the array parameters ( <code>options.collections</code> , <code>options.collections.keys</code> , or <code>options.locales</code> ) is not an array.
SSS_MISSING_REQD_ARGUMENT	A collection or key parameter is missing.
INVALID_TRANSLATION_KEY	The format of a specified key is invalid.
INVALID_TRANSLATION_COLLECTION	The format of a specified collection is invalid.
INVALID_LOCALE	The format of a specified locale is invalid.
INVALID_ALIAS	The format of a specified alias is invalid.

## Syntax



**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 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)

<b>Method Description</b>	Creates a <a href="#">translation.Handle</a> object in the specified locale from an existing <a href="#">translation.Handle</a> object.  This method returns a <a href="#">translation.Handle</a> object that contains the same translation strings as the <code>options.handle</code> object, and the strings are in the <code>options.locale</code> locale. Before you can use this method to select a locale, the locale must be loaded using the <code>locales</code> parameter of <a href="#">translation.load(options)</a> .
<b>Returns</b>	<a href="#">translation.Handle</a>
<b>Supported Script Types</b>	Client and server-side scripts  For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/translation Module</a>
<b>Sibling Object Members</b>	<a href="#">N/translation Module Members</a>
<b>Since</b>	2019.1

## Parameters

Parameter	Type	Required / Optional	Description
<code>options.handle</code>	<a href="#">translation.Handle</a>	required	The <a href="#">translation.Handle</a> object to select a locale for.
<code>options.locale</code>	string	required	The locale to select.  Use the values in the <a href="#">translation.Locale</a> enum to set this value.

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A handle or locale parameter is missing.
WRONG_PARAMETER_TYPE	The <code>options.handle</code> parameter is not a <a href="#">translation.Handle</a> object.
INVALID_LOCALE	The specified <a href="#">translation.Handle</a> object uses an unknown or unsupported locale.
TRANSLATION_HANDLE_IS_IN_AN_ILLEGAL_STATE	The specified <a href="#">translation.Handle</a> object is in an illegal state.

## Syntax

 **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 germanStrings = translation.load({
```

```

collections: [{  
    alias: 'myCollection',  
    collection: 'custcollection_my_strings',  
    keys: ['MY_TITLE', 'MY_MESSAGE'],  
},  
    locales: [translation.Locale.de_DE, translation.Locale.es_ES]  
});  
  
var spanishStrings = translation.selectLocale({  
    handle: germanStrings,  
    locale: translation.Locale.es_ES  
});  
...  
// Add additional code

```

## translation.Locale

<b>Enum Description</b>	Holds the string values for supported locales for Translation Collections. This enum is used to pass the locale argument to <a href="#">translation.get(options)</a> , <a href="#">translation.selectLocale(options)</a> , and <a href="#">translation.load(options)</a> .  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.
<b>Type</b>	enum
<b>Supported Script Types</b>	Client and server-side scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/translation Module</a>
<b>Sibling Object Members</b>	<a href="#">N/translation Module Members</a>
<b>Since</b>	2019.1

## Values

The following table lists all possible locale values and the respective languages. Typically, only some of these locales will be enabled for a company and available to use with Translation Collections.

Locale	Language
CURRENT	Language currently set by the user
COMPANY_DEFAULT	Default company language
af_ZA	Afrikaans (South Africa)
ar	Arabic

bg_BG	Bulgarian (Bulgaria)
bn_BD	Bengali (Bangladesh)
bs_BA	Bosnian (Bosnia and Herzegovina)
cs_CZ	Czech (Czech Republic)
da_DK	Danish (Denmark)
de_DE	German (Germany)
el_GR	Greek (Greece)
en	English
en_AU	English (Australia)
en_CA	English (Canada)
en_GB	English (United Kingdom)
en_US	English (United States)
es_AR	Spanish (Argentina)
es_ES	Spanish (Spain)
et_EE	Estonian (Estonia)
fa_IR	Farsi (Iran)
fi_FI	Finnish (Finland)
fr_CA	French (Canada)
fr_FR	French (France)
gu_IN	Gujarati (India)
he_IL	Hebrew (Israel)
hi_IN	Hindi (India)
hr_HR	Croatian (Croatia)
hu_HU	Hungarian (Hungary)
hy_AM	Armenian (Armenia)
id_ID	Indonesian (Indonesia)
is_IS	Icelandic (Iceland)
it_IT	Italian (Italy)
ja_JP	Japanese (Japan)
kn_IN	Kannada (India)
ko_KR	Korean (Korea)
lb_LU	Luxembourgish (Luxembourg)
lt_LT	Lithuanian (Lithuania)

lv_LV	Latvian (Latvia)
mr_IN	Marathi (India)
ms_MY	Malay (Malaysia)
nl_NL	Dutch (Netherlands)
no_NO	Norwegian (Norway)
pa_IN	Punjabi (India)
pl_PL	Polish (Poland)
pt_BR	Portuguese (Brazil)
pt_PT	Portuguese (Portugal)
ro_RO	Romanian (Romania)
ru_RU	Russian (Russia)
sh_RS	Serbian (Latin)
sk_SK	Slovakian (Slovakia)
sl_SI	Slovenian (Slovenia)
sq_AL	Albanian (Albania)
sr_RS	Serbian (Serbia)
sv_SE	Swedish (Sweden)
ta_IN	Tamil (India)
te_IN	Telugu (India)
th_TH	Thai (Thailand)
tl_PH	Tagalog (Philippines)
tr_TR	Turkish (Turkey)
uk_UA	Ukrainian (Ukraine)
vi_VN	Vietnamese (Viet Nam)
zh_CN	Chinese (Simplified)
zh_TW	Chinese (Traditional)

## Syntax



**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',
        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 Modules

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

Modules within the N/ui namespace allow you to build a custom UI using SuiteScript 2.0. Use `['N/ui']` as the first argument of the define function as a shortcut to load the three modules (N/ui/dialog, N/ui/message, and N/ui/serverWidget) on server scripts, or the N/ui/dialog and N/ui/message on client scripts.

- [N/ui/dialog Module](#) — Used to create modal dialog boxes that can be used to present additional options or alerts. This module uses JavaScript promises to manage dialogs asynchronously.
- [N/ui/message Module](#) — Used to display and manage messages at the top of your User Interface.
- [N/ui/serverWidget Module](#) — Contains the UI components used to work with user interfaces in NetSuite. This module is used to create custom pages, forms, lists and widgets that have the NetSuite look-and-feel.

## N/ui/dialog Module

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

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

Member Type	Name	Property Type / Method Return Type	Supported Script Types	Description
Method	<a href="#">dialog.alert(options)</a>	Promise	Client scripts	Creates an Alert dialog with an OK button.
	<a href="#">dialog.confirm(options)</a>	Promise	Client scripts	Creates a Confirm dialog with OK and Cancel buttons.

Member Type	Name	Property Type / Method Return Type	Supported Script Types	Description
	<a href="#">dialog.create(options)</a>	Promise	Client scripts	Creates a dialog with specified buttons.

## 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](#).

### Sample 1: Create an Alert dialog

- i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).
- i 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. For more information on debugging SuiteScript client scripts, see the help topic [Debugging Client Scripts](#).

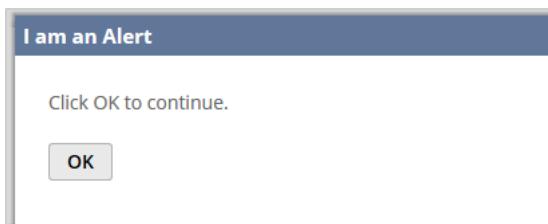
The following sample shows how to create an Alert dialog:

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

require(['N/ui/dialog'],
    function(dialog) {
        var options = {
            title: "I am an Alert",
            message: "Click OK to continue."
        };
        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 screenshot shows the Alert dialog created in this example:



## Sample 2: Create a Confirmation dialog

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

**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. For more information on debugging SuiteScript client scripts, see the help topic [Debugging Client Scripts](#).

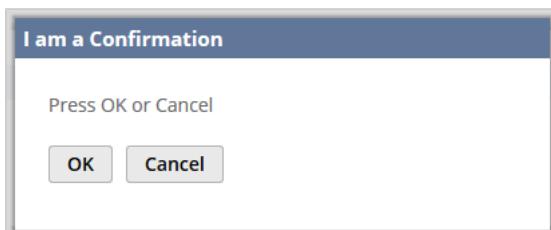
The following sample shows how to create a Confirmation dialog:

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

require(['N/ui/dialog'],
    function(dialog) {
        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);
    });
});
```

The following screenshot shows the Confirmation dialog created in this example:



## Sample 3: Create a dialog with custom buttons

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

**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. For more information on debugging SuiteScript client scripts, see the help topic [Debugging Client Scripts](#).

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);
    });

```

The following screenshot shows the dialog with buttons created in this example:



## Sample 4: Create a dialog with the default button

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

**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. For more information on debugging SuiteScript client scripts, see the help topic [Debugging Client Scripts](#).

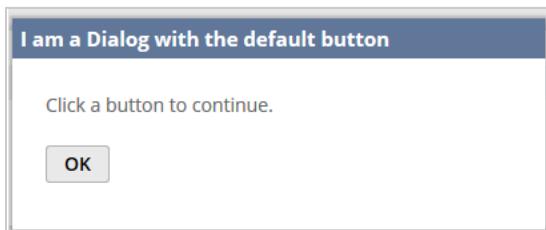
The following sample shows the default behavior when you create a dialog without specifying any buttons, a single button with the label OK.

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

require(['N/ui/dialog'],
    function(dialog) {
        var options = {
            title: 'I am a Dialog with the default button',
            message: 'Click a button to continue.',
        };

        function success(result) {
            console.log("Success with value " + result);
        }
        function failure(reason) {
            console.log("Failure: " + reason);
        }
        dialog.create(options).then(success).catch(failure);
    });
});
```

The following screenshot shows the dialog with the default button created in this example:



## dialog.alert(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates an Alert dialog with an OK button.
<b>Returns</b>	<b>Promise Object.</b> To run a callback function when the OK button is clicked, pass a function to the <b>then</b> portion of the Promise object. When the OK button is clicked, <b>true</b> 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.
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/dialog Module</a>
<b>Since</b>	2016.1

## Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.title	string	optional	The alert dialog title. This value defaults to an empty string.	2016.1
options.message	string	optional	The content of the alert dialog. This value defaults to an empty string.	2016.1

## Syntax

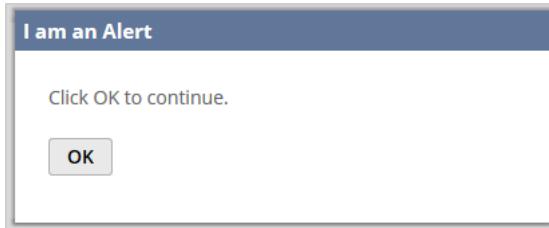


**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/dialog Module Script Samples](#).

```
//Add additional code
...
function success(result) { console.log('Success with value: ' + result) }
function failure(reason) { console.log('Failure: ' + reason) }

dialog.alert({
    title: 'I am an Alert',
    message: 'Click OK to continue.'
}).then(success).catch(failure);
...
//Add additional code
```

The following screenshot shows an example of an Alert dialog:



## dialog.confirm(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a Confirm dialog with OK and Cancel buttons.
<b>Returns</b>	<b>Promise Object.</b> To run a callback function when the OK button is pressed, pass a function to the <b>then</b> portion of the Promise object. The value of the pressed button, where OK is <b>true</b> and Cancel is <b>false</b> , is passed to the callback.
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/dialog Module</a>
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.title	string	optional	The confirmation dialog title. This value defaults to an empty string.	2016.1
options.message	string	optional	The content of the confirmation dialog. This value defaults to an empty string.	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/ui/dialog Module Script Samples](#).

```
//Add additional code
...
var options = {
    title: "I am a Confirmation",
    message: "Press OK or Cancel"
};

function success(result) {
    console.log("Success with value " + result);
}
```

```

}

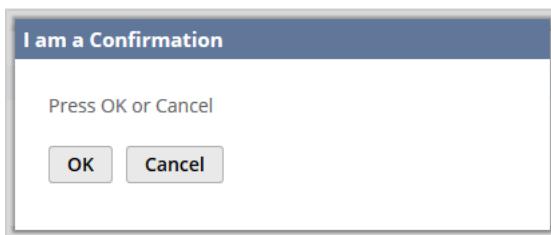
function failure(reason) {
    console.log("Failure: " + reason);
}

dialog.confirm(options).then(success).catch(failure);

...
//Add additional code

```

The following screenshot shows an example of a Confirmation dialog:



## dialog.create(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a dialog with specified buttons.
<b>Returns</b>	<a href="#">Promise Object</a> . To run a callback function when a button is pressed, pass a function to the <code>then</code> portion of the Promise object. The value of the button pressed is passed to the callback.
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/dialog Module</a>
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.buttons</code>	<code>string[]</code>	optional	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.	2016.1

Parameter	Type	Required / Optional	Description	Since
options.title	string	optional	The dialog title. This value defaults to an empty string.	2016.1
options.message	string	optional	The content of the dialog. This value defaults to an empty string.	2016.1

## Errors

Error Code	Error Message	Thrown If
WRONG_PARAMETER_TYPE	Wrong parameter type: {1} is expected as {2}.	The options.buttons parameter is specified and is not an array.
BUTTONS_MUST_INCLUDE_BOTH_A_LABEL_AND_VALUE	Buttons must include both a label and value.	The options.buttons parameter is specified and one or more items do not have a label, a value, or both.

## Syntax

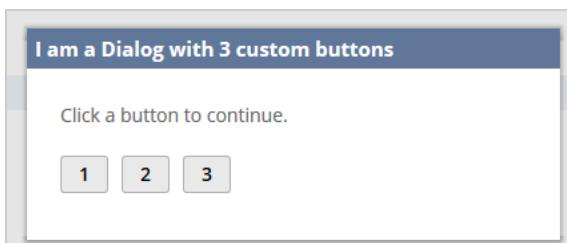
**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/dialog Module Script Samples](#).

```
//Add additional code
...
var options = {
  title: 'I am a Dialog with 3 custom buttons',
  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
```

The following screenshot shows an example of a Custom dialog with three buttons:



If no buttons are specified, a single value with the label OK is used:

```
//Add additional code
...
dialog.create({
    title: 'I am a Dialog with the default button',
    message: 'Click a button to continue.'
});
...
//Add additional code
```



## N/ui/message Module

**Note:** The content in this help topic pertains to SuiteScript 2.0.

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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">message.Message</a>	void	Client scripts	Encapsulates the Message object that gets created when calling the create method.
Method	<a href="#">message.create(options)</a>	<a href="#">message.Message</a>	Client scripts	Creates a message that can be displayed or hidden near the top of the page.
Enum	<a href="#">message.Type</a>	enum	Client scripts	Indicates the type of message to display, which specifies the background color of the message and other message indicators.

## Message Object Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Message.hide()	void	Client scripts	Hides the message.
	Message.show()	void	Client scripts	Shows the message.

## 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](#).

- Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).
- 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. For more information on debugging SuiteScript client scripts, see the help topic [Debugging Client Scripts](#).

The following sample shows how to create messages for the four available types (confirmation, information, warning, and error).

```
/*
 * @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
    });

    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();
    setTimeout(myMsg3.hide, 15000); // will disappear after 15s
  }
);
```

```

        message: "My Message 3",
        type: message.Type.WARNING,
        duration: 20000
    });

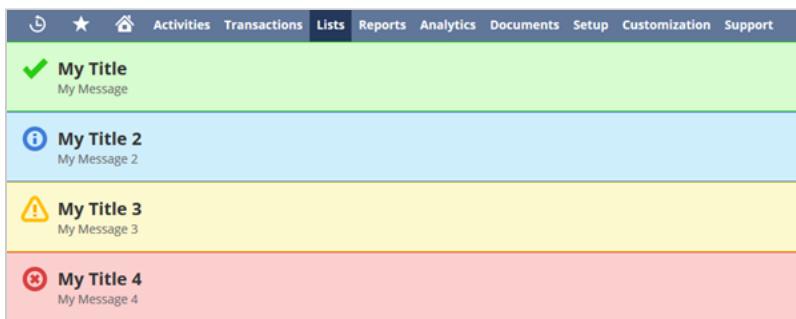
    myMsg3.show(); // will disappear after 20s

    var myMsg4 = message.create({
        title: "My Title 4",
        message: "My Message 4",
        type: message.Type.ERROR
    });

    myMsg4.show(); // will stay up until hide is called.
});

```

You can see the outcome in the following screenshot:



## message.Message

<b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.	
<b>Object Description</b>	The Message object that gets created when calling the create method. For a complete list of this object's methods, see <a href="#">Message Object Members</a> .
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Module</b>	<a href="#">N/ui/message Module</a>
<b>Since</b>	2016.1

## Message.hide()

<b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.	
<b>Method Description</b>	Hides the message.
<b>Returns</b>	void
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .

<b>Governance</b>	None
<b>Module</b>	N/ui/message Module
<b>Since</b>	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/ui/message Module Script Sample](#).

```
//Add additional code
...
var myMsg = message.create({
    title: "My Title 2",
    message: "My Message 2",
    type: message.Type.INFORMATION
});
myMsg.show();
setTimeout(myMsg.hide(), 15000); // hide the message after 15s
...
//Add additional code
```

## Message.show()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Shows the message.
<b>Returns</b>	void
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	N/ui/message Module
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.duration	int string	optional	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 use a string, it will be parsed to a number.  If you specify a duration for message.create() and message.show(), the value from the message.show() method call takes precedence.	2016.1

## Errors

Error Code	Error Message	Thrown If
WRONG_PARAMETER_TYPE	Wrong parameter type: {1} is expected as {2}.	The <code>options.duration</code> is specified with a non-numerical 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/message Module Script Sample](#).

```
//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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a message that can be displayed or hidden near the top of the page.
<b>Returns</b>	<code>message.Message</code> .
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/message Module</a>
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.type</code>	<code>message.Type</code>	required	The message type. Set this value using the <code>message.Type</code> enum.	2016.1
<code>options.title</code>	string	optional	The message title. This value defaults to an empty string.	2016.1
<code>options.message</code>	string	optional	The content of the message. This value defaults to an empty string.	2016.1

Parameter	Type	Required / Optional	Description	Since
options.duration	int string	optional	<p>The amount of time, in milliseconds, to show the message. The default is 0, which shows the message until <code>Message.hide()</code> is called. If you use a string, it will be parsed to a number.</p> <p>If you specify a duration for <code>message.create()</code> and <code>message.show()</code>, the value from the <code>message.show()</code> method call takes precedence.</p>	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/ui/message Module Script Sample](#).

```
//Add additional code
...
var myMsg = message.create({
    title: "My Title",
    message: "My Message",
    type: message.Type.CONFIRMATION
});
...
//Add additional code
```

## message.Type

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Indicates the type of message to display, which specifies the background color of the message and other message indicators.
	 <b>Note:</b> JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation uses the term enumeration (or enum) to describe a plain JavaScript object with a flat, map-like structure. In this object, each key points to a read-only string value.
<b>Supported Script Types</b>	Client scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a> .
<b>Module</b>	<a href="#">N/ui/message Module</a>
<b>Since</b>	2016.1

## Values

Value	Color
CONFIRMATION	A green background with a checkmark icon.

Value	Color
INFORMATION	A blue background with an Information icon.
WARNING	A yellow background with a Warning icon.
ERROR	A red background with an X icon.

## Syntax

 **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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

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](#)
- [Button Object Members](#)
- [Field Object Members](#)
- [FieldGroup Object Members](#)
- [Form Object Members](#)
- [List Object Members](#)
- [ListColumn Object Members](#)
- [Sublist Object Members](#)
- [Tab Object Members](#)

- N/ui/serverWidget Module Script Samples

## N/ui/serverWidget Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">serverWidget.Assistant</a>	Object	Suitelets and beforeLoad user events	Encapsulates a scriptable, multi-step NetSuite assistant.
	<a href="#">serverWidget.AssistantStep</a>	Object	Suitelets and beforeLoad user events	Encapsulates a step within a custom NetSuite assistant.
	<a href="#">serverWidget.Button</a>	Object	Suitelets and beforeLoad user events	Encapsulates a button that appears in a UI object.
	<a href="#">serverWidget.Field</a>	Object	Suitelets and beforeLoad user events	Encapsulates a NetSuite field.
	<a href="#">serverWidget.FieldGroup</a>	Object	Suitelets and beforeLoad user events	Encapsulates a field group.
	<a href="#">serverWidget.Form</a>	Object	Suitelets and beforeLoad user events	Encapsulates a NetSuite form.
	<a href="#">serverWidget.List</a>	Object	Suitelets and beforeLoad user events	Encapsulates a list.
	<a href="#">serverWidget.ListColumn</a>	Object	Suitelets and beforeLoad user events	Encapsulates list columns.
	<a href="#">serverWidget.Sublist</a>	Object	Suitelets and beforeLoad user events	Encapsulates a NetSuite sublist.
	<a href="#">serverWidget.Tab</a>	Object	Suitelets and beforeLoad user events	Encapsulates NetSuite tabs and subtabs.
Method	<a href="#">serverWidget.createAssistant(options)</a>	<a href="#">serverWidget.Assistant</a>	Suitelets and beforeLoad user events	Creates and returns a new assistant object.
	<a href="#">serverWidget.createForm(options)</a>	<a href="#">serverWidget.Form</a>	Suitelets and beforeLoad user events	Creates and returns a new form object.
	<a href="#">serverWidget.createList(options)</a>	<a href="#">serverWidget.List</a>	Suitelets and beforeLoad user events	Instantiates a List object (specifying the title, and whether to hide the navigation bar)
Enum	<a href="#">serverWidget.AssistantSubmitAction</a>	string (read-only)	Suitelets and beforeLoad user events	Holds the string values for submit actions performed by the user.
	<a href="#">serverWidget.FieldBreakType</a>	string (read-only)	Suitelets and beforeLoad user events	Holds the string values for supported field break types. This enum is used to set the value of the <a href="#">Field.updateBreakType(options)</a> property.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	serverWidget.FieldDisplayType	string (read-only)	Suitelets and beforeLoad user events	Holds the string values for supported field display types. This enum is used to set the value of the <a href="#">Field.updateDisplayType(options)</a> property.
	serverWidget.FieldLayoutType	string (read-only)	Suitelets and beforeLoad user events	Holds the string values for the supported types of field layouts. This enum is used to set the value of the <a href="#">Field.updateLayoutType(options)</a> property.
	serverWidgetFieldType	string (read-only)	Suitelets and beforeLoad user events	Holds the values for supported field types. This enum is used to set the value of the <a href="#">Field.type</a> property.
	serverWidgetFormPageLinkType	string (read-only)	Suitelets and beforeLoad user events	Holds the string values for supported page link types on a form. This enum is used to set the value of the <a href="#">type</a> parameter for <a href="#">Form.addPageLink(options)</a> .
	serverWidgetLayoutJustification	string (read-only)	Suitelets and beforeLoad user events	Holds the string values for supported justification layouts. This enum is used to set the value of the align parameter when <a href="#">List.addColumn(options)</a> is called.
	serverWidgetListStyle	string (read-only)	Suitelets and beforeLoad user events	Holds the string values for supported list styles. This enum is used to set the value of the <a href="#">List.style</a> property.
	serverWidgetSublistDisplayType	string (read-only)	Suitelets and beforeLoad user events	Holds the string values for supported sublist display types. This enum is used to set the value of the <a href="#">Sublist.displayType</a> property.
	serverWidgetSublistType	string (read-only)	Suitelets and beforeLoad user events	Holds the string values for valid sublist types. This enum is used to define the <a href="#">type</a> parameter when <a href="#">Form.addSublist(options)</a> is called.

## Assistant Object Members

The following members are called on the [serverWidget.Assistant](#) object.

Member Type	Name	Property Type / Method Return Type	Supported Script Types	Description
Method	Assistant.addField(options)	<a href="#">serverWidget.Field</a>	Suitelets and beforeLoad user events	Adds a field to an assistant.
	Assistant.addFieldGroup(options)	<a href="#">serverWidget.FieldGroup</a>	Suitelets and beforeLoad user events	Adds a field group to an assistant.
	Assistant.addStep(options)	<a href="#">serverWidget.AssistantStep</a>	Suitelets and beforeLoad user events	Adds a step to an assistant.
	Assistant.addSublist(options)	<a href="#">serverWidget.Sublist</a>	Suitelets and beforeLoad user events	Adds a sublist to an assistant.

Member Type	Name	Property Type / Method Return Type	Supported Script Types	Description
	Assistant.getField(options)	serverWidget.Field	Suitelets and beforeLoad user events	Gets a field object.
	Assistant.getFieldGroup(options)	serverWidget.FieldGroup	Suitelets and beforeLoad user events	Gets a field group object.
	Assistant.getFieldGroupIds()	string	Suitelets and beforeLoad user events	Gets all the field group IDs in an assistant.
	Assistant.getFieldIds()	string	Suitelets and beforeLoad user events	Gets all the field IDs in an assistant.
	Assistant.getFieldIdsByFieldGroup(fieldGroup)	string[]	Suitelets and beforeLoad user events	Gets all field IDs in the assistant field group.
	Assistant.getLastAction()	string	Suitelets and beforeLoad user events	Gets the last action submitted by the user.
	Assistant.getLastStep()	serverWidget.AssistantStep	Suitelets and beforeLoad user events	Gets the step that the last submitted action came from.
	Assistant.getNextStep()	serverWidget.AssistantStep	Suitelets and beforeLoad user events	Gets the next step prompted by the assistant.
	Assistant.getStep(options)	serverWidget.AssistantStep	Suitelets and beforeLoad user events	Returns a step in an assistant.
	Assistant.getStepCount()	serverWidget.AssistantStep	Suitelets and beforeLoad user events	Gets the total count of steps in the assistant.
	Assistant.getSteps()	serverWidget.AssistantStep[]	Suitelets and beforeLoad user events	Gets all the steps in the assistant.
	Assistant.getSublist(options)	serverWidget.Sublist	Suitelets and beforeLoad user events	Get a Sublist object from its ID.
	Assistant.getSublistIds()	string	Suitelets and beforeLoad user events	Gets all the sublist IDs in an assistant.
	Assistant.hasErrorHtml()	boolean true   false	Suitelets and beforeLoad user events	Indicates whether the assistant threw an error.
	Assistant.isFinished()	boolean true   false	Suitelets and beforeLoad user events	Sets the status of the assistant. If set to true, the assistant is finished.
	Assistant.sendRedirect(options)	void	Suitelets and beforeLoad user events	Manages redirects in an assistant.

Member Type	Name	Property Type / Method Return Type	Supported Script Types	Description
	Assistant.setSplash(options)	void	Suitelets and beforeLoad user events	Define a splash message.
	Assistant.updateDefaultValues(values)	void	Suitelets and beforeLoad user events	Sets the default values of an array of fields that are specific to the assistant.
Property	Assistant.clientScriptFileDialog	number	Suitelets and beforeLoad user events	The file cabinet ID of client script file to be used in this assistant.
	Assistant.clientScriptModulePath	string	Suitelets and beforeLoad user events	The relative path to the client script file to be used in this assistant.
	Assistant.currentStep	serverWidget.AssistantStep	Suitelets and beforeLoad user events	Identifies the current step.
	Assistant.errorHtml	string	Suitelets and beforeLoad user events	The error message text.
	Assistant.finishedHtml	string	Suitelets and beforeLoad user events	The text displayed after an assistant is finished.
	Assistant.hideAddToShortcutsLink	boolean true   false	Suitelets and beforeLoad user events	Indicates whether the Add to Shortcuts Link is displayed in the UI.
	Assistant.hideStepNumber	boolean true   false	Suitelets and beforeLoad user events	Indicates whether the current and total step numbers are displayed in the UI.
	Assistant.isNotOrdered	boolean true   false	Suitelets and beforeLoad user events	Indicates whether assistant steps are ordered or unordered.
	Assistant.title	string	Suitelets and beforeLoad user events	The title of an assistant.

## AssistantStep Object Members

The following members are called on the [serverWidget.AssistantStep](#) object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	AssistantStep.getFieldIds()	string	Suitelets and beforeLoad user events	Gets all the field IDs in an assistant step.
	AssistantStep.getLineCount(options)	number	Suitelets and beforeLoad user events	Gets the number of lines previously entered by a user in a step.
	AssistantStep.getLineCount(options)	string	Suitelets and beforeLoad user events	Gets all the field IDs in a list.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	AssistantStep.getSubmittedSublistIds()	string	Suitelets and beforeLoad user events	Gets the IDs for all the sublist fields (line items) in a step.
	AssistantStep.getSublistValue(options)	string	Suitelets and beforeLoad user events	Gets the current value of a sublist field (line item) in a step.
	AssistantStep.getValue(options)	string	Suitelets and beforeLoad user events	Gets the current value of a field.
Property	AssistantStep.helpText	string	Suitelets and beforeLoad user events	The help text for a step.
	AssistantStep.id	string	Suitelets and beforeLoad user events	The internal ID of the step.
	AssistantStep.label	string	Suitelets and beforeLoad user events	The label for a step.
	AssistantStep.stepNumber	number	Suitelets and beforeLoad user events	Indicates where this step appears sequentially in an assistant.

## Button Object Members

The following members are called on the [serverWidget.Button](#) object.

Member Type	Name	Property Type	Supported Script Types	Description
Property	Button.isDisabled	boolean true   false	Suitelets and beforeLoad user events	Indicates whether a button is grayed-out and disabled.
	Button.isHidden	boolean true   false	Suitelets and beforeLoad user events	Indicates whether the button is hidden in the UI.
	Button.label	string	Suitelets and beforeLoad user events	The label for the button.

## Field Object Members

The following members are called on the [serverWidget.Field](#) object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Field.addSelectOption(options)	void	Suitelets and beforeLoad user events	Adds a select option to a dropdown list for a selectable field.
	Field.getSelectOptions(options)	object[]	Suitelets and beforeLoad user events	Returns the internal ID and label of the options for a

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
				select field as name/value pairs.
	Field.setHelpText(options)	void	Suitelets and beforeLoad user events	Sets the help text that appears in the field help popup.
	Field.updateBreakType(options)	serverWidget.Field	Suitelets and beforeLoad user events	Updates the break type used to add a break in flow layout for the field.
	Field.updateDisplaySize(options)	serverWidget.Field	Suitelets and beforeLoad user events	Updates the height and width for the field.
	Field.updateDisplayType(options)	serverWidget.Field	Suitelets and beforeLoad user events	Updates the type of display for the field.
	Field.updateLayoutType(options)	serverWidget.Field	Suitelets and beforeLoad user events	Updates the layout type for the field.
Property	Field.alias	string	Suitelets and beforeLoad user events	The alias used to set the field value.
	Field.defaultValue	string	Suitelets and beforeLoad user events	The default value for the field.
	Field.id	string	Suitelets and beforeLoad user events	The internal ID for the field.
	Field.isMandatory	boolean true   false	Suitelets and beforeLoad user events	Indicates whether the field is mandatory.
	Field.label	string	Suitelets and beforeLoad user events	Gets or sets the label for the field.
	Field.linkText	string	Suitelets and beforeLoad user events	The text displayed for a link in place of the URL.
	Field.maxLength	number	Suitelets and beforeLoad user events	The maximum length, in characters, for the field.
	Field.padding	number	Suitelets and beforeLoad user events	The number of empty vertical character spaces above the field.
	Field.richTextHeight	number	Suitelets and beforeLoad user events	The height of a rich text field, in pixels.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	Field.richTextBoxWidth	number	Suitelets and beforeLoad user events	The width of a rich text field, in pixels.
	Field.type	string	Suitelets and beforeLoad user events	The type of field.

## FieldGroup Object Members

The following members are called on the [serverWidget.FieldGroup](#) object.

Member Type	Name	Property Type	Supported Script Types	Description
Property	FieldGroup.isBorderHidden	boolean <code>true</code>   <code>false</code>	Suitelets and beforeLoad user events	Indicates whether a border appears around the field group.
	FieldGroup.isCollapsible	boolean <code>true</code>   <code>false</code>	Suitelets and beforeLoad user events	Indicates whether the field group is collapsible.
	FieldGroup.isCollapsed	boolean <code>true</code>   <code>false</code>	Suitelets and beforeLoad user events	Indicates whether the field group is initially collapsed or expanded in the default view.
	FieldGroup.isSingleColumn	boolean <code>true</code>   <code>false</code>	Suitelets and beforeLoad user events	Indicates whether the field group is aligned.
	FieldGroup.label	string	Suitelets and beforeLoad user events	The label for the field group.

## Form Object Members

The following members are called on the [serverWidget.Form](#) object.

Member Type	Name	Property Type / Method Return Type	Supported Script Types	Description
Method	Form.addButton(options)	<a href="#">serverWidget.Button</a>	Suitelets and beforeLoad user events	Adds a button to the form.
	Form.addCredentialField(options)	<a href="#">serverWidget.Field</a>	Suitelets and beforeLoad user events	Adds a field that store credentials in NetSuite for invoking services provided by third parties.
	Form.addField(options)	<a href="#">serverWidget.Field</a>	Suitelets and beforeLoad user events	Adds a field to the form.
	Form.addFieldGroup(options)	<a href="#">serverWidget.FieldGroup</a>	Suitelets and beforeLoad user events	Adds a group of fields to the form.

Member Type	Name	Property Type / Method Return Type	Supported Script Types	Description
	Form.addPageInitMessage(options)	void	Suitelets and beforeLoad user events	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.
	Form.addPageLink(options)	void	Suitelets and beforeLoad user events	Adds a link to a form.
	Form.addResetButton(options)	serverWidget.Button	Suitelets and beforeLoad user events	Adds a reset button to a form that clears user input.
	Form.addSecretKeyField(options)	serverWidget.Field	Suitelets and beforeLoad user events	Add a secret key field to the form.
	Form.addSublist(options)	serverWidget.Sublist	Suitelets and beforeLoad user events	Adds a sublist to the form.
	Form.addSubmitButton(options)	serverWidget.Button	Suitelets and beforeLoad user events	Adds a submit button to a form that saves user inputs.
	Form.addTab(options)	serverWidget.Tab	Suitelets and beforeLoad user events	Adds a subtab to a form.
	Form.addTab(options)	serverWidget.Tab	Suitelets and beforeLoad user events	Adds a tab to a form.
	Form.getButton(options)	serverWidget.Button	Suitelets and beforeLoad user events	Returns a button by internal ID.
	Form.getField(options)	serverWidget.Field	Suitelets and beforeLoad user events	Returns a field by internal ID.
	Form.getSublist(options)	serverWidget.Sublist	Suitelets and beforeLoad user events	Returns a sublist by internal ID.
	Form.getSubtab(options)	serverWidget.Tab	Suitelets and beforeLoad user events	Returns a subtab by internal ID.
	Form.getTab(options)	serverWidget.Tab	Suitelets and beforeLoad user events	Returns a tab object from its internal ID.
	Form.getTabs()	serverWidget.Tab[]	Suitelets and beforeLoad user events	Returns an array of all the tabs in a form.
	Form.insertField(options)	void	Suitelets and beforeLoad user events	Inserts a field before another field within a form.
	Form.insertSublist(options)	serverWidget.Sublist	Suitelets and beforeLoad user events	Inserts a sublist before another sublist on a form.

Member Type	Name	Property Type / Method Return Type	Supported Script Types	Description
	Form.insertSubtab(options)	serverWidget.Tab	Suitelets and beforeLoad user events	Inserts a subtab before another subtab on a form.
	Form.insertTab(options)	serverWidget.Tab	Suitelets and beforeLoad user events	Inserts a tab before another tab on a form.
	Form.removeButton(options)	void	Suitelets and beforeLoad user events	Removes a button from a form.
	Form.updateDefaultValues(options)	void	Suitelets and beforeLoad user events	Sets the default values of many fields on a form.
Property	Form.clientScriptFileDialog	number	Suitelets and beforeLoad user events	The file cabinet ID of client script file to be used in this form.
	Form.clientScriptModulePath	string	Suitelets and beforeLoad user events	The relative path to the client script file to be used in this form.
	Form.title	string	Suitelets and beforeLoad user events	The title used for the form.

## List Object Members

The following members are called on the [serverWidget.List](#) object.

Member Type	Name	Property Type / Method Return Type	Supported Script Types	Description
Method	List.addButton(options)	serverWidget.Button	Suitelets and beforeLoad user events	Adds a button to a list.
	List.addColumn(options)	serverWidget.ListColumn	Suitelets and beforeLoad user events	Adds a column to a list.
	List.addEditColumn(options)	serverWidget.ListColumn	Suitelets and beforeLoad user events	Adds a column containing Edit or Edit/View links to a Suitelet or Portlet list.
	List.addPageLink(options)	void	Suitelets and beforeLoad user events	Adds a link to a list.
	List.addRow(options)	void	Suitelets and beforeLoad user events	Adds a single row to a list.
	List.addRows(options)	void	Suitelets and beforeLoad user events	Adds multiple rows to a list.

Member Type	Name	Property Type / Method Return Type	Supported Script Types	Description
Property	List.clientScriptFileDialog	number	Suitelets and beforeLoad user events	The file cabinet ID of client script file to be used in this list.
	List.clientScriptModulePath	string	Suitelets and beforeLoad user events	The relative path to the client script file to be used in this list.
	List.style	string	Suitelets and beforeLoad user events	Sets the display style for this list.
	List.title	string	Suitelets and beforeLoad user events	Sets the List title.

## ListColumn Object Members

The following members are called on the [serverWidget.ListColumn](#) object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	ListColumn.addParamToURL(options)	serverWidget.ListColumn	Suitelets and beforeLoad user events	Adds a URL parameter (optionally defined per row) to the list column's URL.
	ListColumn.setURL(options)	serverWidget.ListColumn	Suitelets and beforeLoad user events	Sets the base URL for the list column.
Property	ListColumn.label	string	Suitelets and beforeLoad user events	The label of this list column.

## Sublist Object Members

The following members are called on the [serverWidget.Sublist](#) object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Sublist.addButton(options)	serverWidget.Button	Suitelets and beforeLoad user events	Adds a button to a sublist.
	Sublist.addField(options)	serverWidget.Field	Suitelets and beforeLoad user events	Add a field to a sublist.
	Sublist.addMarkAllButtons()	serverWidget.Button	Suitelets and beforeLoad user events	Adds a Mark All or Unmark All button.
	Sublist.addRefreshButton()	serverWidget.Button	Suitelets and beforeLoad user events	Adds a Reset button.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	Sublist.getField(options)	serverWidget.Field	Suitelets and beforeLoad user events	Returns a Field object on a specified sublist.
	Sublist.getSublistValue(options)	string	Suitelets and beforeLoad user events	Gets a field value on a sublist.
	Sublist.setSublistValue(options)	void	Suitelets and beforeLoad user events	Sets the value of a sublist field.
	Sublist.updateTotallingFieldId(options)	serverWidget.Sublist	Suitelets and beforeLoad user events	Updates the ID of a field designated as a totalling column, which is used to calculate and display a running total for the sublist.
	Sublist.updateUniqueFieldId(options)	serverWidget.Sublist	Suitelets and beforeLoad user events	Updates a field ID that is to have unique values across the rows in the sublist.
Property	Sublist.displayType	string	Suitelets and beforeLoad user events	The display style for a sublist.
	Sublist.helpText	string	Suitelets and beforeLoad user events	The inline help text for a sublist.
	Sublist.label	string	Suitelets and beforeLoad user events	The label for a sublist.
	Sublist.lineCount	number (read-only)	Suitelets and beforeLoad user events	The number of line items in a sublist.

## Tab Object Members

The following members are called on the `serverWidget.Tab` object.

Member Type	Name	Value Type	Supported Script Types	Description
Property	Tab.helpText	string	Suitelets and beforeLoad user events	The inline help text for a tab or subtab.
	Tab.label	string	Suitelets and beforeLoad user events	The label for a tab or subtab.

## 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](#).

## Sample 1: Create a Suitelet that generates a sample form

**i Note:** This script sample uses the `define` function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the `require` function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample creates a Suitelet that generates a sample form with a submit button, fields, and an inline editor sublist.

```
/*
 * @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'
    });
    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
};
});
}

```

## Sample 2: Create a Suitelet that generates a customer survey form

**i Note:** This script sample uses the `define` function, which is required for an entry point script (a script you attach to a script record and deploy). You must use the `require` function if you want to copy the script into the SuiteScript Debugger and test it. For more information, see [SuiteScript 2.0 Global Objects](#).

The following sample 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>";

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><br><br>";

form.addField({
    id:'custpage_lblproductrating',
    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: 'custpage_rdoprodproductrating',
    type: serverWidget.FieldType.RADIO,
    label: '1',
    source: 'p1'
}).updateLayoutType({
    layoutType: serverWidget.FieldLayoutType.STARTROW
});
form.addField({
    id: 'custpage_rdoprodproductrating',
    type: serverWidget.FieldType.RADIO,
    label: '2',
    source: 'p2'
}).updateLayoutType({
    layoutType: serverWidget.FieldLayoutType.MIDROW
});
form.addField({
    id: 'custpage_rdoprodproductrating',
    type: serverWidget.FieldType.RADIO,

```

```

        label: '3',
        source: 'p3'
    }).updateLayoutType({
        layoutType: serverWidget.FieldLayoutType.MIDROW
    });
    form.addField({
        id: 'custpage_rdoproductrating',
        type: serverWidget.FieldType.RADIO,
        label: '4',
        source: 'p4'
    }).updateLayoutType({
        layoutType: serverWidget.FieldLayoutType.MIDROW
    });
    form.addField({
        id: 'custpage_rdoproductrating',
        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
    });

    form.addSubmitButton({
        label: 'Submit'
    });
    form.addResetButton({
        label: 'Reset'
    });

    context.response.writePage(form);
}
return {
    onRequest: onRequest
}
});

```

## serverWidget.Assistant

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	<p>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.</p> <p>All data and states for an assistant are tracked automatically throughout the user's session until completion of the assistant.</p> <p>You can create a new assistant with the <a href="#">serverWidget.createAssistant(options)</a> method.</p> <p>After you create an <b>Assistant</b> object, you can:</p> <ul style="list-style-type: none"> <li>■ Build and run an assistant in your NetSuite account.</li> <li>■ Add a variety of scriptable elements to the assistant including fields, steps, buttons, tabs, and sublists.</li> </ul> <p>For a complete list of this object's methods and properties, see <a href="#">Assistant Object Members</a>.</p>
<b>Supported Script Types</b>	<p>SuiteScript 2.0 Suitelet Script Type and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a></p>
<b>Module</b>	<p>N/ui/serverWidget Module</p>

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'
});
...
//Add additional code
```

## Assistant.addField(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a field to an assistant. Use fields to record or display information specific to your needs.
<b>Returns</b>	serverWidget.Field object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	The internal ID for this field.	2015.2
options.label	string	required	The label for this field.	2015.2
options.type	string	required	The field type. Use the <a href="#">serverWidget.FieldType</a> enum to set this value.	2015.2

Parameter	Type	Required / Optional	Description	Since
			<p><b>Note:</b> If you have set the <code>type</code> parameter to <code>SELECT</code>, and you want to add custom options to the select field, you must set <code>source</code> to <code>NULL</code>. Then, when a value is specified, the value will populate the options from the source.</p> <p><b>Important:</b> 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.</p>	
<code>options.source</code>	string	optional	<p>The internalId or scriptId of the source list for this field. Use this parameter if you are adding a select (List/Record) or multi-select type of field.</p> <p>For radio fields only, the <code>source</code> parameter is not an optional parameter, it must contain the radio button's unique internal ID. The <code>id</code> parameter contains the ID that identifies all the radio buttons of the same group.</p> <p><b>Note:</b> If you want to add custom options on a select field, you must set the source parameter to <code>NULL</code>, and then add the custom options using <code>Field.addSelectOption(options)</code>.</p> <p><b>Important:</b> After you create a select or multi-select field that is sourced from a record or list, you cannot add additional values with <code>Field.addSelectOption(options)</code>. The select values are determined by the source record or list.</p>	2015.2
<code>options.container</code>	string	optional	The internal ID of the field group to place this field in.	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/ui/serverWidget Module Script Samples](#).

```
//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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">FieldGroup.isCollapsed</a> and <a href="#">FieldGroup.isCollapsible</a> properties.
<b>Returns</b>	<a href="#">serverWidget.FieldGroup</a> object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <a href="#">beforeLoad(scriptContext)</a> )
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
<code>options.id</code>	string	required	The internal ID for the field group.	2015.2
<code>options.label</code>	string	required	The label for the field group.	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'
});
assistant.addFieldGroup({
    id : 'idname',
    label : 'Sample label'
});
...
//Add additional code
```

## Assistant.addStep(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a step to an assistant. Steps define each page of the assistant.  Use <a href="#">Assistant.isNotOrdered</a> 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 <a href="#">AssistantStep.helpText</a> on the object returned.
<b>Returns</b>	<code>serverWidget.AssistantStep</code> object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
<code>options.id</code>	string	required	The internal ID for this step (for example, 'entercontacts').	2015.2
<code>options.label</code>	string	required	The label for this step (for example, 'Enter Contacts'). By default, the step appears vertically in the left panel of the assistant.	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'
});
assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
...
//Add additional code
```

## Assistant.addSublist(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a sublist to an assistant.
	<b>Note:</b> Only inline editor sublists are added. Other sublist types are not supported.
<b>Returns</b>	<code>serverWidget.Sublist</code> object
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type</a> ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
<code>options.id</code>	string	required	The internal ID for the sublist.	2015.2
<code>options.label</code>	string	required	The label for the sublist.	2015.2
<code>options.type</code>	string	required	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 <a href="#">serverWidget.SublistType</a> .	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/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.INLINEEDITOR
});
...
//Add additional code
```

## Assistant.getField(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a field object on an assistant page.
<b>Returns</b>	<code>serverWidget.Field</code>
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.id</code>	string	required	The internal ID of the field.	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'
});
assistant.addField({
    id : 'idname',
    type : serverWidget.FieldType.TEXT,
    label : 'Sample label'
});
var field = assistant.getField({
    id: 'idname'
});
...
//Add additional code
```

## Assistant.getFieldGroup(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a field group on an assistant page.
---------------------------	---

<b>Returns</b>	serverWidget.FieldGroup object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	The internal ID of the field group.	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'
});
assistant.addFieldGroup({
    id : 'idname',
    label : 'Sample label'
});
var fieldgroup = assistant.getFieldGroup({
    id: 'idname'
});
...
//Add additional code
```

## Assistant.getFieldGroupIds()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Retrieves all the internal IDs for field groups in an assistant.
<b>Returns</b>	string[]
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None

<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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'
});
assistant.addFieldGroup({
    id : 'idname',
    label : 'Sample label'
});
var fieldgroupid = assistant.getFieldGroupIds();
...
//Add additional code
```

## Assistant.getFieldIds()

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets all the internal IDs for fields in an assistant.
<b>Returns</b>	string[]
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type</a> (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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'
});
assistant.addField({
```

```

        id : 'idname',
        label : 'Sample label'
    });
var fieldid = assistant.getFieldIds();
...
//Add additional code

```

## Assistant.getFieldIdsByFieldGroup(fieldGroup)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets all field IDs in the assistant field group.
<b>Returns</b>	string[]
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2016.1

### Parameters

Parameter	Type	Required / Optional	Description	Since
fieldGroup	string	required	The internal ID of the field group.	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/ui/serverWidget Module Script Samples](#).

```

//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addFieldGroup({
    id : 'fieldgroupid',
    label : 'Sample label'
});
assistant.addField({
    id : 'idname',
    label : 'Sample label'
});
var fieldid = assistant.getFieldIdsByFieldGroup({
    fieldGroup : 'fieldgroupid'
});
...

```

```
//Add additional code
```

## Assistant.getLastAction()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the last action taken by the user. To identify the step that the last action came from, use <a href="#">Assistant.getLastStep()</a> .
<b>Returns</b>	string
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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'
});
...
if (assistant.getLastAction() == serverWidget.AssistantSubmitAction.CANCEL) {
    ...
}
...

//Add additional code
```

## Assistant.getLastStep()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the step that the last submitted action came from.
<b>Returns</b>	A <a href="#">serverWidget.AssistantStep</a> object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>

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 lastStep = assistant.getLastStep();
...
//Add additional code
```

## Assistant.getNextStep()

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the next step corresponding to the user's last submitted action in the assistant. If you need information about the last step, use <a href="#">Assistant.getLastStep()</a> before you use this method.
<b>Returns</b>	<a href="#">serverWidget.AssistantStep</a> object
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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 nextStep = assistant.getNextStep();
...
//Add additional code
```

## Assistant.getStep(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a step in an assistant.
<b>Returns</b>	<code>serverWidget.AssistantStep</code> object
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type</a> ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.id</code>	string	required	The internal ID of the step.	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'
});
assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
var firststep = assistant.getStep({
    id: 'idname'
});
...
//Add additional code
```

## Assistant.getStepCount()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the total number of steps in an assistant.
---------------------------	---

<b>Returns</b>	The total count of assistant steps as a number
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type</a> (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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'
});
assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
var numSteps = assistant.getStepCount();
...
//Add additional code
```

## Assistant.getSteps()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets all the steps in an assistant.
<b>Returns</b>	<a href="#">serverWidget.AssistantStep[]</a> object
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type</a> (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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'
});
assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
var steps = assistant.getSteps();
...
//Add additional code

```

## Assistant.getSublist(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a sublist in an assistant.
<b>Returns</b>	serverWidget.Sublist object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	The internal ID of the sublist.	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'
});
assistant.addSublist({
    id : 'idname',
    label : 'Sample label',
    type : serverWidget.SublistType.LIST
});
var sublist = assistant.getSublist({

```

```

        id : 'idname'
});
...
//Add additional code

```

## Assistant.getSublistIds()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the IDs for all the sublists in an assistant.
<b>Returns</b>	string[]
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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'
});
assistant.addSublist({
    id : 'idname',
    label : 'Sample label',
    type : serverWidget.SublistType.LIST
});
var sublistid = assistant.getSublistIds();
...
//Add additional code

```

## Assistant.hasErrorHtml()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Determine whether an assistant has an error message to display for the current step.
<b>Returns</b>	boolean <code>true</code> if <code>Assistant.errorHtml</code> contains a value.
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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'
});
...
if (assistant.hasErrorHtml()) {
    ...
}
...
//Add additional code
```

## Assistant.isFinished()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Indicates whether all steps in an assistant are completed. If set to <b>true</b> , the assistant is finished and a completion message displays. To set the text for the completion message, use the <a href="#">Assistant.finishedHtml</a> property.
<b>Returns</b>	boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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'
});
```

```

...
if (assistant.isFinished()) {
    ...
}
...

//Add additional code

```

## Assistant.sendRedirect(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	void
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.response	response	required	The response that redirects the user.	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: '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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Defines a splash message.
<b>Returns</b>	void
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.title	string	required	The title of the splash screen.	2015.2
options.text1	string	required	Text for the splash screen	2015.2
options.text2	string	optional	Text for a second column on the splash screen, if desired.	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](#).

```

...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.setSplash({
    title: "Welcome Title!",
    text1: "An explanation of what this assistant accomplishes.",
    text2: "Some parting words."
});
...

```

```
//Add additional code
```

## Assistant.updateDefaultValues(values)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the default values of an array of fields that are specific to the assistant.
<b>Returns</b>	void
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type</a> (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2016.1

### Parameters

Parameter	Type	Required / Optional	Description	Since
values	object[]	required	An array of fields to update.	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'
});
assistant.addField({
    id : 'idname',
    type : serverWidget.FieldType.TEXT,
    label : 'Sample label'
});
assistant.updateDefaultValues({
    idname : "New Default Value"
});
...
//Add additional code
```

## Assistant.clientScriptFileDialog

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The file cabinet ID of client script file to be used in this assistant.
-----------------------------	---

Type	number
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

Error Code	Thrown If
PROPERTY_VALUE_CONFLICT	You attempted to set this value when the Assistant.clientScriptModulePath property value has already been specified. For more information, see <a href="#">Assistant.clientScriptModulePath</a> .

## Syntax

 **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.clientscriptId = 32;
...
//Add additional code
```

## Assistant.clientScriptModulePath

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

Property Description	The relative path to the client script file to be used in this assistant.
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	2016.2

## Errors

Error Code	Thrown If
PROPERTY_VALUE_CONFLICT	You attempted to set this value when the Assistant.clientScriptModulePath property value has already been specified. For more information, see <a href="#">Assistant.clientScriptModulePath</a> .

## Syntax



**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
...
objAssistant.clientScriptModulePath = 'SuiteScripts/assistantBehavior.js';
...
//Add additional code
```

## Assistant.currentStep



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Identifies the current step.  You can set any step as the current step.
<b>Type</b>	<code>serverWidget.AssistantStep</code> (read-only)
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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'
});
assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
var step = assistant.currentStep;
...
//Add additional code
```

## Assistant.errorHtml



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Error message text for the current step.
-----------------------------	--

	Optionally, you can use HTML tags to format the message.
<b>Type</b>	string
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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'
});
assistant.errorHtml = "You have <b>not</b> filled out the required fields. Please go back.";
...
//Add additional code
```

## Assistant.finishedHtml

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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 <a href="#">Assistant.isFinished()</a> .
<b>Type</b>	string
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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'
});
assistant.finishedHtml = "Congratulations! You have successfully set up your account.";
...
```

```
//Add additional code
```

## Assistant.hideAddToShortcutsLink



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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 <b>false</b> , which means the Add to Shortcuts link is visible in the UI.  If set to <b>true</b> , the Add To Shortcuts link is not visible on an Assistant page.
<b>Type</b>	boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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'
});
assistant.hideAddToShortcutsLink = true;
...
//Add additional code
```

## Assistant.hideStepNumber



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether assistant steps are displayed with numbers.  By default, the value is <b>false</b> , which means that steps are numbered.  If set to <b>true</b> , the assistant does not use step numbers.  To change step ordering, set <a href="#">Assistant.isNotOrdered</a> .
<b>Type</b>	boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>

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'
});
assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
assistant.hideStepNumber = true;
...
//Add additional code
```

## Assistant.isNotOrdered

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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 <b>false</b> , which means the steps are ordered. Ordered steps appear vertically in the left panel of the assistant  If set to <b>true</b> , steps can be completed in any order. In the UI, unordered steps appear horizontally and below the assistant title
<b>Type</b>	boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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'
});
assistant.addStep({
    id : 'idname',
```

```

        label : 'Sample label'
});
assistant.addStep({
    id : 'idname2',
    label : 'Sample label 2'
});
assistant.isNotOrdered = false;
...
//Add additional code

```

## Assistant.title



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The title for the assistant. The title appears at the top of all assistant pages. This value overrides the title specified in <a href="#">serverWidget.createAssistant(options)</a> .
<b>Type</b>	string
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type</a> ( <a href="#">beforeLoad(scriptContext)</a> )
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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 title = assistant.title;
...
//Add additional code

```

## serverWidget.AssistantStep



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a step within a custom NetSuite assistant. Create a step by calling <a href="#">Assistant.addStep(options)</a> . For a complete list of this object's methods and properties, see <a href="#">AssistantStep Object Members</a> .
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type</a> ( <a href="#">beforeLoad(scriptContext)</a> )

<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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'
});
...
//Add additional code
```

## AssistantStep.getFieldIds()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the IDs for all the fields in a step.
<b>Returns</b>	string[]
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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'
});
assistant.addField({
```

```

        id : 'fieldid',
        type : serverWidget.FieldType.TEXT,
        label : 'Field'
});
var fieldIds = assistantStep.getFieldIds();
...
//Add additional code

```

## AssistantStep.getSublistFieldIds(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the IDs for all the sublist fields (line items) in a step.
<b>Returns</b>	string[]
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.group	string	required	The sublist internal ID.	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 sublistFieldIds = assistantStep.getSublistFieldIds({

```

```

        group : 'sublistid'
});
...
//Add additional code

```

## AssistantStep.getLineCount(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the number of lines on a sublist in a step.
	<b>i Note:</b> The first line number on a sublist is 0 (not 1).
<b>Returns</b>	The count of line items on a sublist as a number
	<b>i Note:</b> if the sublist does not exist, -1 is returned.
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.group	string	required	The sublist internal ID.	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 numLines = assistantStep.getLineCount({
    group : 'sublistid'
});
...
//Add additional code

```

## AssistantStep.getSubmittedSublistIds()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the IDs for all the sublists submitted in a step.
<b>Returns</b>	string[]
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the current value of a sublist field (line item) in a step.
---------------------------	--

<b>Returns</b>	The value of a sublist field as a string
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type</a> ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.group</code>	string	required	The internal ID of the sublist.	2015.2
<code>options.id</code>	string	required	The internal ID of the sublist field.	2015.2
<code>options.line</code>	number	required	The line number for the sublist field.	2015.2

 **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](#).

```
//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)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets the current value(s) of a field or multi-select field.
<b>Returns</b>	string[]
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	The internal ID of a field.	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 field = assistant.addField({  
    id : 'fieldid',  
    type : serverWidget.FieldType.TEXT,  
    label : 'Text'  
});  
var value = assistantStep.getValue({  
    id: 'fieldid'  
});  
...  
//Add additional code
```

## AssistantStep.helpText



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The help text for a step.
<b>Type</b>	string
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 ...
assistantStep.helpText = 'Help Text Goes Here.';
...
//Add additional code
```

## AssistantStep.id



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal ID of the step.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 id = assistantStep.id';
```

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

## AssistantStep.label

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The label for the step.
	 <b>Note:</b> To create a label when the step is first added to the assistant, you can use the <a href="#">Assistant.addStep(options)</a> method.
<b>Type</b>	string
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 label = assistantStep.label;
...
//Add additional code
```

## AssistantStep.stepNumber

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates where this step appears sequentially in the assistant.
<b>Type</b>	The index of this step as a number.
	 <b>Note:</b> A sequence of assistant steps starts at 1.
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Module</b>	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 stepNum = assistantStep.stepNumber;
...
//Add additional code
```

## serverWidget.Button

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates button that appears in a UI object.  To add a button, use <a href="#">Form.addButton(options)</a> or <a href="#">Sublist.addButton(options)</a> . When adding a button to a record or form, consider using a <code>beforeLoad</code> user event script.  Custom buttons only appear during Edit mode. On records, custom buttons appear to the left of the printer icon.
	<b>Note:</b> 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 <a href="#">Configuring Buttons and Actions</a> .
	For a complete list of this object's properties, see <a href="#">Button Object Members</a> .
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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 button = form.addButton({
    id : 'buttonid',
    label : 'Test'
});
...
//Add additional code
```

## Button.isDisabled



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>Indicates whether a button is grayed-out and disabled. The default value is <b>false</b>. If set to true, the button appears grayed-out in the and cannot be clicked.</p> <p><b>Note:</b> This method is not supported for standard NetSuite buttons. This method can be used with custom buttons only.</p>
<b>Type</b>	boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 button = form.addButton({
    id : 'buttonid',
    label : 'Test'
});
button.isDisabled = true;
...
//Add additional code
```

## Button.isHidden



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether the button is hidden in the UI.
-----------------------------	---

	<p>The default value is <code>false</code>, which means the button is visible.</p> <p>If set to true, the button is not visible in the UI.</p> <p><b>Note:</b> This property is supported on custom buttons and on some standard NetSuite buttons. For a list of supported standard buttons, see the help topic <a href="#">Button IDs</a>.</p>
<b>Type</b>	<code>boolean true   false</code>
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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 button = form.addButton({
    id : 'buttonid',
    label : 'Test'
});
button.isHidden = true;
...
//Add additional code
```

## Button.label



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>The label for the button.</p> <p>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.</p> <p><b>Note:</b> This property is supported on custom buttons and most standard buttons.</p>
<b>Type</b>	<code>string</code>
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>

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 button = form.addButton({
    id : 'buttonid',
    label : 'Test'
});
var label = button.label;
...
//Add additional code
```

## serverWidget.Field

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a body or sublist field. Use fields to record or display information specific to your needs.  To add a Field object, use <a href="#">Assistant.addField(options)</a> , <a href="#">Form.addField(options)</a> , or <a href="#">Sublist.addField(options)</a> .  For a complete list of this object's methods and properties, see <a href="#">Field Object Members</a> .
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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 field = form.addField({
    id : 'custpage_text',
    type : serverWidget.FieldType.TEXT,
```

```

        label : 'Text'
});
...
//Add additional code

```

## Field.addSelectOption(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds the select options that appears in the dropdown of a field.
	<b>Important:</b> After you create a select or multi-select field that is sourced from a record or list, you cannot add additional values with <code>Field.addSelectOption(options)</code> . The select values are determined by the source record or list.
<b>Returns</b>	void
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
<code>options.value</code>	string	required	The internal ID of this select option.	2015.2
<code>options.text</code>	string	required	The label for this select option.	2015.2
<code>options.isSelected</code>	boolean <code>true</code>   <code>false</code>	optional	If set to <code>true</code> , this option is selected by default in the UI. The default value for this parameter is <code>false</code> .	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 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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	Object[]
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.filter	string	optional	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.	2015.2

Parameter	Type	Required / Optional	Description	Since
			<p>"John Smith" or "Shauna Johnson", only those 10 options will be returned.</p> <p>Filter values are case insensitive. The filters 'John' and 'john' will return the same select options.</p>	
options.filteroperator	string	optional	<p>Supported operators are <code>contains</code>   <code>is</code>   <code>startswith</code>.</p> <p>If not specified, defaults to the <code>contains</code> operator.</p>	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 selectField = form.addField({
    id : 'custpage_selectfield',
    type : serverWidget.FieldType.SELECT,
    label : 'Select'
});
selectField.addSelectOption({
    value : 'a',
    text : 'Albert'
});
var options = field.getSelectOptions({
    filter : 'a',
    filteroperator: 'startswith'
});
...
//Add additional code
```

## Field.setHelpText(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the help text for the field.  When the field label is clicked, a popup displays the help text defined using this method.
<b>Returns</b>	The <code>serverWidget.Field</code> object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>

<b>Since</b>	2015.2
--------------	--------

## Parameters

<b>i Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
options.help	string	required	The text in the field help popup.	2015.2
options.showInlineForAssistant	boolean true   false	optional	If set to <b>true</b> , the field help will display inline below the field on the assistant, and in a field help popup.  The default value is <b>false</b> , which means the field help appears in a popup when the field label is clicked and does not appear inline.	2015.2

**i Note:** The **inline** parameter is available only to [serverWidget.Field](#) objects that have been added to [serverWidget.createAssistant\(options\)](#) objects.

## Syntax

<b>⚠ Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/ui/serverWidget Module Script Samples</a> .
---

```
//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.updateBreakType(options)

<b>i Note:</b> The content in this help topic pertains to SuiteScript 2.0.
--

<b>Method Description</b>	Updates the break type used to add a break in flow layout for the field.
<b>Returns</b>	<a href="#">serverWidget.Field</a> object

<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.breakType	serverWidget.FieldBreakType	required	The break type of the field.	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 field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateBreakType({
    breakType : serverWidget.FieldBreakType.STARTCOL
});
...
//Add additional code
```

## Field.updateDisplaySize(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">Field.richTextWidth</a> and <a href="#">Field.richTextHeight</a> .
<b>Returns</b>	<a href="#">serverWidget.Field</a> object

<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2016.1

## Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.height	number	required	The new height of the field.	2015.2
options.width	number	required	The new width of the field.	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 field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateDisplaySize({
    height : 60,
    width : 100
});
...
//Add additional code
```

## Field.updateDisplayType(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Updates the display type for the field.
<b>Returns</b>	serverWidget.Field object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.displayType	string	required	The new display type of the field. For more information about possible values, see <a href="#">serverWidget.FieldDisplayType</a> .	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 field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateDisplayType({
    displayType : serverWidget.FieldDisplayType.HIDDEN
});
...
//Add additional code
```

## Field.updateLayoutType(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Updates the layout type for the field.
<b>Returns</b>	<a href="#">serverWidget.Field</a> object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module

<b>Since</b>	2016.1
--------------	--------

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.layoutType	serverWidget.FieldLayoutType	required	The new layout type of the field.	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 field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateLayoutType({
    layoutType : serverWidget.FieldLayoutType.NORMAL
});
...
//Add additional code
```

## Field.alias

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	An alternate name that you can assign to a <a href="#">serverWidget.Field</a> object.  By default, the alias is equal to the field's internal ID.  This property is only supported on scripted fields created using the <a href="#">N/ui/serverWidget Module</a> .
<b>Type</b>	string
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <a href="#">beforeLoad(scriptContext)</a> )
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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 field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.alias = 'fieldid';
...
//Add additional code
```

## Field.defaultValue

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The default value for this field.  If you pass an empty string or any value that is not a number, such as <code>undefined</code> , the field defaults to a blank field in the UI.  This property is supported only on scripted fields created using the <a href="#">N/ui/serverWidget Module</a> .
<b>Type</b>	string
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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 field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.defaultValue = 'Insert Text Here.';
```

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

## Field.id



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The field internal ID.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
var fieldId = field.id;
...
//Add additional code
```

## Field.isMandatory



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether the field is mandatory or optional. If set to <code>true</code> , then the field is defined as mandatory. The default value is <code>false</code> . This property is supported only on scripted fields created using the <a href="#">N/ui/serverWidget Module</a> .
<b>Type</b>	boolean <code>true</code>   <code>false</code>
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Module</b>	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 field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.isMandatory = true;
...
//Add additional code
```

## Field.label

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The field label. There is a 40-character limit for custom field labels.
<b>Type</b>	string
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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 field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
var label = field.label;
```

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

## Field.linkText



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The text displayed for a link in place of the URL.
<b>Type</b>	string
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.URL,
    label : 'URL'
});
field.linkText = 'NetSuite';
...
//Add additional code
```

## Field.maxLength



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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 <a href="#">N/ui/serverWidget Module</a> .
<b>Type</b>	number
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.maxLength = 64;
...
//Add additional code
```

## Field.padding

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The number of empty vertical character spaces above the field. This property is supported only on scripted fields created using the <a href="#">N/ui/serverWidget Module</a> .
<b>Type</b>	number
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.padding = 50;
...
//Add additional code
```

## Field.richTextHeight



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The height of a rich text field, in pixels. The minimum value is 100 pixels and the maximum value is 500 pixels.
	<b>Note:</b> Rich Text Editing must be enabled.
<b>Type</b>	number
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.RICHTEXT,
    label : 'Rich Text'
});
field.richTextHeight = 50;
...
//Add additional code
```

## Field.richTextWidth



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The width of a rich text field, in pixels. The minimum value is 250 pixels and the maximum value is 800 pixels.
	<b>Note:</b> Rich Text Editing must be enabled.
<b>Type</b>	number
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.RICHTEXT,
    label : 'Rich Text'
});
field.richTextWidth = 100;
...
//Add additional code
```

## Field.type

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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 <a href="#">format.Type</a> .  The maximum character limit for select field types is 801.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 field = form.addField({
    id : 'custpage_textfield',
```

```

        type : serverWidget.FieldType.TEXT,
        label : 'Text'
    });
    var fieldtype = field.type;
    ...
//Add additional code

```

## serverWidget.FieldGroup

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a field group on <a href="#">serverWidget.createAssistant(options)</a> objects and on <a href="#">serverWidget.Form</a> 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 <a href="#">FieldGroup Object Members</a> .
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>(beforeLoad(scriptContext))</code> )
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether the field group can be collapsed.
-----------------------------	---

	If set to <b>false</b> , a border around the field group is displayed. The default value is <b>false</b> .
<b>Type</b>	boolean <b>true   false</b>
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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'
});
fieldgroup.isBorderHidden = true;
...
//Add additional code
```

## FieldGroup.isCollapsible

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether the field group can be collapsed.  The default value is <b>false</b> , which means the field group displays as a static group that cannot be opened or closed.  If set to true, the field group can be collapsed.  Only supported for fields on <a href="#">serverWidget.createAssistant(options)</a> objects
<b>Type</b>	boolean <b>true   false</b>
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Module</b>	N/ui/serverWidget 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/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'
});
fieldgroup.isCollapsible = true;
...
//Add additional code
```

## FieldGroup.isCollapsed



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether field group is collapsed or expanded.  The default value is <code>false</code> , 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 <code>serverWidget.createAssistant(options)</code> objects
<b>Type</b>	boolean <code>true</code>   <code>false</code>
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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 collapsed = fieldgroup.isCollapsed;
...
//Add additional code

```

## FieldGroup.isSingleColumn



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Indicates whether the field group is aligned. The default value is <b>false</b> .
<b>Type</b>	boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>(beforeLoad(scriptContext))</code> )
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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;
...
//Add additional code

```

## FieldGroup.label



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The label for the field group.
<b>Type</b>	string
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 label = fieldgroup.label;
...
//Add additional code
```

## serverWidget.Form



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	<p>Encapsulates a NetSuite-looking form</p> <p>After you create a <b>Form</b> object, you can:</p> <ul style="list-style-type: none"> <li>■ Add a variety of scriptable elements to the form including fields, links, buttons, tabs, and sublists.</li> </ul> <p>For a complete list of this object's methods and properties, see <a href="#">Form Object Members</a>.</p>
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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'
});
...
//Add additional code
```

## Form.addButton(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a button to a form.
<b>Returns</b>	serverWidget.Button object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

## Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.id	string	optional	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 <b>custpage</b> . For example, if you add a button that appears as <b>Update Order</b> , the button internal ID should be something similar to <b>custpage_updateorder</b> .	2015.2
options.label	string	required	The label for this button.	2015.2
options.functionName	string	optional	The function name to be triggered on a click event.	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/ui/serverWidget Module Script Samples](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});

form.addButton({
    id : 'buttonid',
    label : 'Test'
});
...
//Add additional code
```

## Form.addCredentialField(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	<p>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.</p> <p>For example, when executing credit card transactions, merchants need to store credentials in NetSuite that are used to communicate with Payment Gateway providers.</p> <p>The credentials added with this method can be used with the <a href="#">N/sftp Module</a> and the <a href="#">N/https Module</a>.</p> <p>Note the following about this method:</p> <ul style="list-style-type: none"> <li>▪ Credentials associated with this field are stored in encrypted form.</li> <li>▪ No piece of SuiteScript holds a credential in clear text mode.</li> <li>▪ NetSuite reports or forms will never provide to the end user the clear text form of a credential.</li> <li>▪ Any exchange of the clear text version of a credential with a third party must occur over SSL.</li> <li>▪ 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).</li> <li>▪ Decryption occurs through the scripts listed in the <code>restrictToScriptIds</code> parameter. These scripts can call <code>https.createSecureString(options)</code> to decrypt the GUID and create a SecureString instance.</li> </ul> <p> <b>Important:</b> The default maximum length for a credential field is 32 characters. If needed, use the <code>Field.maxLength</code> property to change this value.</p>
<b>Returns</b>	<code>serverWidget.Field</code> object
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Governance</b>	None

<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.id</code>	string	required	The internal ID of the credential field.  The internal ID must be in lowercase, contain no spaces, and include the prefix <code>custpage</code> if you are adding the field to an existing page.	2015.2
<code>options.label</code>	string	required	The label for the credential field.	2015.2
<code>options.restrictToDomains</code>	string   string[]	required	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.	2015.2
<code>options.restrictToScriptIds</code>	string   string[]	required	The IDs of the scripts that are allowed to use this credential field. For example, 'customscript_my_script'.  Scripts defined here can call <a href="#">https.createSecureString(options)</a> to decrypt the GUID.	2015.2
<code>options.restrictToCurrentUser</code>	boolean <code>true</code>   <code>false</code>	optional	Controls whether use of this credential is restricted to the same user that originally entered the credential.  By default, the value is <code>false</code> , 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 <code>true</code> , the credentials apply to a single user.	2015.2
<code>options.container</code>	string	optional	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.	2016.1

## 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](#).

```
//Add additional code
...
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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a field to a form.
<b>Returns</b>	<a href="#">serverWidget.Field</a> object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	The internal ID of the 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. For example, if you add a field that appears as <b>Purchase Details</b> , the field internal ID should be something similar to <code>custpage_purchasedetails</code> or <code>custpage_purchase_details</code> .	2015.2
options.label	string	required	The label for this field.	2015.2
options.type	string	required	The field type for the field. Use the <a href="#">serverWidget.FieldType</a> enum to define the field type.	2015.2
 <b>Important:</b> 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.				
options.source	string	optional	The internalId or scriptId of the source list for this field if it is a select (List/Record) or multi-select field.	2015.2

Parameter	Type	Required / Optional	Description	Since
			<p><b>Important:</b> After you create a select or multi-select field that is sourced from a record or list, you cannot add additional values with <code>Field.addSelectOption(options)</code>. The select values are determined by the source record or list.</p> <p><b>Note:</b> For radio fields only, the <code>source</code> parameter must contain the internal ID for the field. For more information about working with radio buttons, see the help topic <a href="#">Working with Radio Buttons</a>.</p>	
<code>options.container</code>	string	optional	<p>The internal ID of the tab or field group to add the field to.</p> <p>By default, the field is added to the main section of the form.</p>	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/ui/serverWidget Module Script Samples](#).

```
//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
```

## Form.addFieldGroup(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a group of fields to a form.
<b>Returns</b>	<code>serverWidget.FieldGroup</code> object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>

<b>Since</b>	2015.2
--------------	--------

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
<code>options.id</code>	string	required	An internal ID for the field group.	2015.2
<code>options.label</code>	string	required	The label for this field group.	2015.2
<code>options.tab</code>	string	optional	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.	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_text',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    container : 'fieldgroupid'
});
...
//Add additional code
```

## Form.addPageInitMessage(options)

<b>Method Description</b>	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 <a href="#">context.UserEventType</a> .
<b>Returns</b>	Void
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	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.

Parameter	Type	Required / Optional	Description	Since
<code>options.message</code>	<code>message.Message</code>	required	Encapsulates the message to be shown on the form.	2018.2

Parameter	Type	Required / Optional	Description	Since
<code>options.type</code>	<code>message.Type</code>	required	The message type.	2016.1
<code>options.title</code>	string	optional	The message title. This value defaults to an empty string.	2016.1
<code>options.message</code>	string	optional	The content of the message. This value defaults to an empty string.	2016.1
<code>options.duration</code>	int	optional	<p>The amount of time, in milliseconds, to show the message. The default is 0, which shows the message until <code>Message.hide()</code> is called.</p> <p>If you specify a duration for <code>message.create()</code> and <code>message.show()</code>, the value from the <code>message.show()</code> method call takes precedence.</p>	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/ui/serverWidget Module Script Samples](#).

```
//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);
    }
    ...
//Add additional code

```

## Form.addPageLink(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a link to a form.
	 <b>Note:</b> You cannot choose where the page link appears.
<b>Returns</b>	void
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.title	string	required	The text label for the link.	2015.2
options.type	string	required	The type of page link to add. Use the <a href="#">serverWidget.FormPageLinkType</a> enum to set the value.	2015.2
options.url	string	required	The URL for the link.	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'
});
form.addPageLink({
    type : serverWidget.FormPageLinkType.CROSSLINK,
    title : 'NetSuite',
    url : 'http://www.netsuite.com'
}

```

```
})
...
//Add additional code
```

## Form.addButton(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a reset button to a form. The reset buttons allows a user to clear the entries.
<b>Returns</b>	<code>serverWidget.Button</code> object
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type</a> ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.label</code>	string	optional	The label used for this button. If no label is provided, the label defaults to <b>Reset</b> .	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'
});
form.addButton({
    label : 'Reset Button'
});
...
//Add additional code
```

## Form.addSecretKeyField(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a secret key field to the form.
---------------------------	--------------------------------------

	This key can be used in crypto modules to perform encryption or hashing.
	<p><b>Important:</b> The default maximum length for a secret key field is 32 characters. If needed, use the <a href="#">Field.maxLength</a> property to change this value.</p>
<b>Returns</b>	<a href="#">serverWidget.Field</a> object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.id</code>	string	required	The internal ID of the secret key field.  The internal ID must be in lowercase, contain no spaces, and include the prefix <code>custpage</code> if you are adding the field to an existing page.	2016.1
<code>options.restrictTo ScriptIds</code>	string or string[]	required	The script ID of the script that is allowed to use this field.	2016.1
<code>options.label</code>	string	required	The UI label for the field.	2016.1
<code>options.restrictTo CurrentUser</code>	boolean <code>true</code>   <code>false</code>	optional	Controls whether use of this secret key is restricted to the same user that originally entered the key.  By default, the value is <code>false</code> , which means that multiple users can use the key.  If set to <code>true</code> , the secret key applies to a single user.	2016.1
<code>options.container</code>	string	optional	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.	2016.1

## 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](#).

```
//Add additional code
...

```

```

var form = serverWidget.createForm({
    title : 'Simple Form'
});
skField = form.addSecretKeyField({
    id : 'password',
    restrictToScriptIds : 'customscript_my_script',
    restrictToCurrentUser : false,
});
skField.maxLength = 64;
...
//Add additional code

```

## Form.addSublist(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Add a sublist to a form.  <b>i Note:</b> If the row count exceeds 25, sorting is not supported on static sublists created using this method.
<b>Returns</b>	A <code>serverWidget.Sublist</code> object
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.id</code>	string	required	The internal ID name of the sublist.  The internal ID must be in lowercase, contain no spaces, and include the prefix <code>custpage</code> if you are adding the sublist to an existing page. For example, if you add a sublist that appears as <b>Purchase Details</b> , the sublist internal ID should be something equivalent to <code>custpage_purchasedetails</code> or <code>custpage_purchase_details</code> .	2015.2
<code>options.label</code>	string	required	The label for this sublist.	2015.2
<code>options.tab</code>	string	optional	The tab under which to display this sublist. If empty, the sublist is added to the main tab.	2015.2
<code>options.type</code>	string	required	The sublist type.  Use the <code>serverWidget.SublistType</code> enum to set the value.	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 sublist = form.addSublist({
    id : 'sublistid',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
...
//Add additional code
```

## Form.addButton(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a submit button to a form.
	<p><b>Note:</b> If the row count exceeds 25, sorting is not supported on static sublists created using this method.</p>
<b>Returns</b>	serverWidget.Button object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.label	string	optional	The label for this button. If no label is provided, the label defaults to "Save".	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/ui/serverWidget Module Script Samples](#).

```
//Add additional code
```

```

...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addSubmitButton({
    label : 'Submit Button'
});
...
//Add additional code

```

## Form.addSubtab(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a subtab to a form.
	<p> <b>Note:</b> 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.</p> <p> <b>Note:</b> 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.</p>
<b>Returns</b>	serverWidget.Tab object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.id</code>	string	required	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 <code>custpage</code> . For example, if you add a subtab that appears as <b>Purchase Details</b> , the subtab internal ID should be something similar to <code>custpage_purchasedetails</code> or <code>custpage_purchase_details</code> .	2015.2
<code>options.label</code>	string	required	The label for this subtab.	2015.2
<code>options.tab</code>	string	optional	The tab under which to display this sublist. If empty, the sublist is added to the main tab.	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'
});
form.addSubtab({
    id : 'subtabid',
    label : 'Subtab'
});
...
//Add additional code
```

## Form.addTab(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a tab to a form.
	<p><b>Note:</b> 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.</p>
	<p><b>Note:</b> 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.</p>
<b>Returns</b>	<code>serverWidget.Tab</code> object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

## Parameters



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

Parameter	Type	Required / Optional	Description	Since
<code>options.id</code>	string	required	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 <code>custpage</code> . For example, if	2015.2

Parameter	Type	Required / Optional	Description	Since
			you add a subtab that appears as <b>Purchase Details</b> , the subtab internal ID should be something similar to <code>custpage_purchasedetails</code> or <code>custpage_purchase_details</code> .	
<code>options.label</code>	string	required	The label for this tab.	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 tab = form.addTab({
    id : 'tabid',
    label : 'Tab'
});
...
//Add additional code
```

## Form.getButton(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a Button object by internal ID.
<b>Returns</b>	<code>serverWidget.Button</code> object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.id</code>	string	required	The internal ID name of the button.  Internal IDs must be in lowercase and contain no spaces.	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 button = form.addButton({
    id : 'buttonid',
    label : 'Test'
});
var button = form.getButton({
    id : 'buttonid'
});
...
//Add additional code
```

## Form.getField(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a Field object by internal ID.
<b>Returns</b>	<code>serverWidget.Field</code> object
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

## Parameters



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

Parameter	Type	Required / Optional	Description	Since
<code>options.id</code>	string	required	The internal ID name of the field.  Internal IDs must be in lowercase and contain no spaces.	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'
});
form.addField({
    id : 'custpage_text',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
var field = form.getField({
    id : 'textfield'
});
...
//Add additional code

```

## Form.getSublist(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a Sublist object by internal ID.
<b>Returns</b>	serverWidget.Sublist object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	The internal ID name of the sublist.  Internal IDs must be in lowercase and contain no spaces.	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'
}

```

```

});
form.addSublist({
    id : 'sublistid',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
var sublist = form.getSublist({
    id : 'sublistid'
});
...
//Add additional code

```

## Form.getSubtab(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a subtab by internal ID.
<b>Returns</b>	serverWidget.Tab object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	The internal ID name of the subtab.  Internal IDs must be in lowercase and contain no spaces.	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'
});
form.addSubtab({
    id : 'subtabid',

```

```

        label : 'Subtab'
});
var subtab = form.getSubtab({
    id : 'subtabid'
});
...
//Add additional code

```

## Form.getTab(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a tab object from its internal ID.
<b>Returns</b>	serverWidget.Tab object.
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2016.1

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	The internal ID name of the tab to retrieve.	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/ui/serverWidget Module Script Samples](#).

```

//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()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns an array that contains all the tabs in a form.
<b>Returns</b>	serverWidget.Tab[] objects
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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'
});
form.addTab({
    id : 'tabid',
    label : 'Tab'
});
var tabs = form.getTabs();
...
//Add additional code
```

## Form.insertField(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Inserts a field in front of another field.
<b>Returns</b>	void
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.field	serverWidget.Field	required	The Field object to insert.	2016.1
options.nextfield	string	required	The internal ID name of the field you are inserting a field in front of.	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 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)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Inserts a sublist in front of another sublist.
<b>Returns</b>	void
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>

<b>Since</b>	2015.2
--------------	--------

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
options.sublist	serverWidget.Sublist	required	The Sublist object to insert.	2016.1
options.nextsublist	string	required	The internal ID name of the sublist you are inserting a sublist in front of.	2015.2

## Syntax

<b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/ui/serverWidget Module Script Samples</a> .
---

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var sublist1 = form.addSublist({
    id : 'sublistid1',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
var sublist2 = form.addSublist({
    id : 'sublistid2',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
form.insertSublist({
    sublist : sublist2,
    nextsublist : 'sublistid1'
});
...
//Add additional code
```

## Form.insertSubtab(options)

<b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.
--

<b>Method Description</b>	Inserts a subtab in front of another subtab.
<b>Returns</b>	void
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>

<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.subtab	serverWidget.Tab	required	The Subtab object to insert.	2016.1
options.nextsubtab	string	required	The internal ID name of the subtab you are inserting a subtab in front of.	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 subtab1 = form.addSubtab({
    id : 'subtabid1',
    label : 'Subtab'
});
var subtab2 = form.addSubtab({
    id : 'subtabid2',
    label : 'Subtab'
});
form.insertSubtab({
    subtab : subtab2,
    nextsubtab : 'subtabid1'
});
...
//Add additional code
```

## Form.insertTab(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Inserts a tab in front of another tab.
<b>Returns</b>	void

<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.tab	serverWidget.Tab	required	The Tab object to insert.	2016.1
options.nexttab	string	required	The internal ID name of the tab you are inserting a tab in front of.	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 tab1 = form.addTab({
    id : 'tabid1',
    label : 'Tab'
});
var tab2 = form.addTab({
    id : 'tabid2',
    label : 'Tab'
});
form.insertTab({
    tab: tab2,
    nexttab:'tabid1'
});
...
//Add additional code
```

## Form.removeButton(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">Button IDs</a> .
<b>Returns</b>	<code>void</code>
<b>Supported Script Types</b>	<p><a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a></p> <p>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</p>
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.id</code>	string	required	<p>The internal ID name of the button to remove. See the help topic <a href="#">Button IDs</a>.</p> <p>The internal ID must be in lowercase and contain no spaces.</p>	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
...
function beforeLoad(context) {
    var yourForm = context.form;
    yourForm.removeButton('refund');
}
...
//Add additional code
```

## Form.updateDefaultValues(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Updates the default values of multiple fields on the form.
<b>Returns</b>	<code>void</code>
<b>Supported Script Types</b>	<p><a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a></p>
<b>Governance</b>	None

<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2016.1

## Parameters

Parameter	Type	Required / Optional	Description	Since
values	object[]	required	An object containing an array of name/value pairs that map field names to field values.	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/ui/serverWidget Module Script Samples](#).

```
//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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The internal file ID of client script file to be used in this form.  Use this property when attaching an on demand client script to a server-side script.
<b>Type</b>	number
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module

<b>Since</b>	2015.2
--------------	--------

## Errors

Error Code	Thrown If
PROPERTY_VALUE_CONFLICT	You attempted to set this value when the Form.clientScriptModulePath property value has already been specified. For more information, see <a href="#">Form.clientScriptModulePath</a> .

## Syntax

**⚠ 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'
});
form.clientScriptFileId = 32;
...
//Add additional code
```

## Form.clientScriptModulePath

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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.
	<b>i Note:</b> If you deploy a client script to a form using Form.clientScriptFileId or Form.clientScriptModulePath, using the <a href="#">N/log Module</a> adds the logs to the deployment of the parent script. The parent script can be either a beforeLoad user event script or a <a href="#">SuiteScript 2.0 Suitelet Script Type</a> .
<b>Type</b>	string
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2016.2

## Errors

Error Code	Thrown If
PROPERTY_VALUE_CONFLICT	You attempted to set this value when the Form.clientScriptFileId property value has already been specified. For more information, see <a href="#">Form.clientScriptFileId</a> .

## Syntax



**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
...
objForm.clientScriptModulePath = 'SuiteScripts/formBehavior.js';
...
//Add additional code
```

## Form.title



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The title used for the form.
<b>Type</b>	string
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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 title = form.title;
...
//Add additional code
```

## serverWidget.List



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a list. For a complete list of this object's methods and properties, see <a href="#">List Object Members</a> .
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>

<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 list = serverWidget.createList({
    title : 'Simple List'
});
...
//Add additional code
```

## List.addButton(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a button to a list.
<b>Returns</b>	serverWidget.Button object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

## Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	The internal ID of the button.  The internal ID must be in lowercase, contain no spaces, and include the prefix <b>custpage</b> if you are adding the button to an existing page. For example, if you add a button that appears as <b>Update Order</b> , the button internal ID should be something similar to <b>custpage_updateorder</b> .	2015.2
options.label	string	required	The label for this button.	2015.2
options.functionName	string	optional	The function name to call when clicking on this button.	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 list = serverWidget.createList({
    title : 'Simple List'
});
list.addButton({
    id : 'buttonid',
    label : 'Test'
});
...
//Add additional code
```

## List.addColumn(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a column to a list.
<b>Returns</b>	serverWidget.ListColumn object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

## Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	The internal ID of this column.  The internal ID must be in lowercase, contain no spaces.	2015.2
options.label	string	required	The label for this column.	2015.2
options.type	string	required	The field type for this column.  <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <span></span> <b>Note:</b> CHECKBOX field types are not supported.         </div> For more information about possible values, see <a href="#">serverWidget.FieldType</a> .	2015.2

Parameter	Type	Required / Optional	Description	Since
options.align	string	optional	The default value is <code>left</code> .  The layout justification for this column. Possible values include <code>center</code> , <code>right</code> , <code>left</code>	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 list = serverWidget.createList({
    title : 'Simple List'
});
list.addColumn({
    id : 'column1',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    align : serverWidget.LayoutJustification.RIGHT
});
...
//Add additional code
```

## List.addEditColumn(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	<code>serverWidget.ListColumn</code> object
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.column	object	required	The Edit/View column is added to the left of the column specified here.	2015.2

Parameter	Type	Required / Optional	Description	Since
options.showHrefCol	boolean true   false	optional	If set to true, the URL for the link is clickable.	2015.2
options.showView	boolean true   false	optional	If true then an <b>Edit/View</b> column will be added. Otherwise only an <b>Edit</b> column will be added.	2015.2
options.link	string	optional	The <b>Edit/View</b> base link. (For example: /app/common/entity/employee.nl)  The complete link is formed like this: <link>?<linkParamName>=<row data from linkParam>. (For example: /app/common/entity/employee.nl?id=123)	2019.2
options.linkParam	string	optional	The internal ID of the field in the row data where to take the parameter from.  The default value is the value set in the options.column parameter.	2019.2
<span style="color: green;">✓</span> <b>Tip:</b> In most cases, the value to use here is <b>internalid</b>				
options.linkParamName	string	optional	The name of the parameter.  The default value is <b>id</b> .	2019.2

## 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 list = serverWidget.createList({
    title : 'Simple List'
});

list.addColumn({
    id: 'internalid',
    type: 'text',
    label: 'Number'
});
list.addColumn({
    id: 'entityid',
    type: 'text',
    label: 'Name'
});
list.addEditColumn({
    column : 'entityid',
    showHrefCol: true,
    showView : true,
    link: '/app/common/entity/employee.nl',
    linkParam: 'internalid',
    linkParamName: 'id',
});
```

```
});  
...  
//Add additional code
```

## List.addPageLink(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a link to a list.
<b>Returns</b>	void
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
options.title	string	required	The text label for the link.	2015.2
options.type	string	required	The type of page link to add.  For more information about possible values, see <a href="#">serverWidget.FormPageLinkType</a> .	2015.2
options.url	string	required	The URL for the link.	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 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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a single row to a list.
<b>Returns</b>	<code>serverWidget.List</code>
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description	Since
<code>options.row</code>	object	required	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.	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 list = serverWidget.createList({
    title : 'Simple List'
});
list.addRow({
    row : { columnid1 : 'value1', columnid2 : 'value2' }
});
...
//Add additional code
```

## List.addRows(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds multiple rows to a list.
---------------------------	-------------------------------

<b>Returns</b>	serverWidget.ListColumn
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.rows	object[]	required	An array of rows that consist of either a search.Result array, or an array of name/value pairs. Each pair should contain the value for the corresponding Column object in the list.	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
...
list.addRows({
    rows : [{columnid1 : 'value1', columnid2 : 'value2'},
             {columnid1 : 'value2', columnid2 : 'value3'}]
});
...
//Add additional code
```

## List.clientScriptFileDialog

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The file cabinet ID of client script file to be used in this list.
<b>Type</b>	number
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2016.1

## Errors

Error Code	Thrown If
PROPERTY_VALUE_CONFLICT	You attempted to set this value when the List.clientScriptModulePath property value has already been specified. For more information, see <a href="#">List.clientScriptModulePath</a> .

## Syntax

**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 list = serverWidget.createList({
    title : 'Simple List'
});
list.clientScriptFileDialog = 123;
...
//Add additional code
```

## List.clientScriptModulePath

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The relative path to the client script file to be used in this list.
<b>Type</b>	string
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2016.2

## Errors

Error Code	Thrown If
PROPERTY_VALUE_CONFLICT	You attempted to set this value when the List.clientScriptFileDialog property value has already been specified. For more information, see <a href="#">List.clientScriptFileDialog</a> .

## Syntax

**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
...
objList.clientScriptModulePath = 'SuiteScripts/listBehavior.js';
...
```

```
//Add additional code
```

## List.style



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Sets the display style for this list. For more information about possible values, see <a href="#">serverWidget.ListStyle</a> .
<b>Type</b>	string
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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 list = serverWidget.createList({
    title : 'Simple List'
});
list.style = serverWidget.ListStyle.REPORT;
...
//Add additional code
```

## List.title



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Sets the list title.
<b>Type</b>	string
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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 list = serverWidget.createList({
    title : 'Simple List'
});
var title = list.title;
...
//Add additional code

```

## serverWidget.ListColumn



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a list column For a complete list of this object's methods and properties, see <a href="#">ListColumn Object Members</a> .
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a URL parameter (optionally defined per row) to the list column's URL.
<b>Returns</b>	<code>serverWidget.ListColumn</code> object

<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description
options.param	string	required	The name for the parameter.
options.value	string	required	The value for the parameter.
options.dynamic	boolean	optional	If true, then the parameter value is actually an alias that is calculated per row.

## Syntax

 **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 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.setURL(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the base URL for the list column.
<b>Returns</b>	<code>serverWidget.ListColumn</code>

<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2016.1

## Parameters

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

Parameter	Type	Required / Optional	Description
options.url	string	required	The base URL or a column in the data source that returns the base URL for each row
options.dynamic	boolean	optional	If true, then the URL is actually an alias that is calculated per row.

## Syntax

 **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 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
```

## ListColumn.label

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	This list column label.
<b>Type</b>	string
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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/ui/serverWidget Module Script Samples](#).

```
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});
var listcolumn = list.addColumn({
    id : 'column1',
    type : serverWidget.FieldType.URL,
    label : 'URL',
});
var label = listcolumn.label;
...
//Add additional code
```

## serverWidget.Sublist



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a sublist on a <a href="#">serverWidget.Form</a> or an <a href="#">serverWidget.createAssistant(options)</a> object.  To add a sublist, use <a href="#">Assistant.addSublist(options)</a> or <a href="#">Form.addSublist(options)</a> .  <b>Note:</b> This object is read-only except for instances created via the <code>serverWidget</code> module using Suitelets or beforeLoad user event scripts.
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
...
//Add additional code

```

## Sublist.addButton(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a button to a sublist.
<b>Returns</b>	<code>serverWidget.Button</code>
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.id</code>	string	required	The internal ID name of the button. The internal ID must be in lowercase and without spaces.
<code>options.label</code>	string	required	The label for the button.
<code>options.functionName</code>	string	optional	The function name to be triggered on a button click.

### Syntax

 **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 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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a field to a sublist.
<b>Returns</b>	serverWidget.Field object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	The internal ID for this field.  The internal ID must be in lowercase and without spaces.	2015.2
options.label	string	required	The label for this field.	2015.2
options.type	string	required	The field type.  Use the <a href="#">serverWidget.FieldType</a> enum to set this value. The INLINEHTML and RICHTEXT values are not supported with this method. The MULTISELECT value is not supported for SuiteScript 2.0 Suitelets.	2015.2
 <b>Note:</b> If you have set the <code>type</code> parameter to <code>SELECT</code> , and you want to add custom options to the select field, you must set <code>source</code> to <code>NULL</code> .				
options.source	string	optional	The internalId or scriptId of the source list for this field.	2015.2

Parameter	Type	Required / Optional	Description	Since
			<p>Use this parameter if you are adding a select (List/Record) type of field.</p> <p><b>Note:</b> If you want to add custom options on a select field, you must set the source parameter to NULL.</p> <p><b>Important:</b> After you create a select or multi-select field that is sourced from a record or list, you cannot add additional values with <code>Field.addSelectOption(options)</code>. The select values are determined by the source record or list.</p>	

## Syntax

**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 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()

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a Mark All and an Unmark All button to a LIST type of sublist.
<b>Returns</b>	A <code>serverWidget.Button[]</code> object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>

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 sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.LIST,
    label : 'List Sublist'
});
sublist.addMarkAllButtons();
...
//Add additional code
```

## Sublist.addButton()

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Adds a Refresh button to a LIST type of sublist.
<b>Returns</b>	<code>serverWidget.Button</code> object
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type</a> ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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 sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.LIST,
    label : 'List Sublist'
```

```

});  
sublist.addRefreshButton();  
...  
//Add additional code

```

## Sublist.getField(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a Field object on a sublist.
<b>Returns</b>	serverWidget.Field
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2016.2

### Parameters

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

Parameter	Type	Required / Optional	Description
options.id	string	required	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 <a href="#">SuiteScript Records Browser</a> .

### Syntax

 **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 itemField = form.getSublist({id: 'item'}).getField({id: 'item'});  
...  
//Add additional code

```

## Sublist.getSublistValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Gets a field value on a sublist.
<b>Returns</b>	string

<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.id	string	required	The internal ID of a field.
options.line	number	required	 <b>Note:</b> The first line number on a sublist is 0 (not 1).

## Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required parameter is not passed.
YOU_CANNOT_CALL_1_METHOD_ON_SUBRECORD_FIELD_SUBLIST_2_FIELD_3	You called {1} method on subrecord field. Sublist: {2}, field: {3}.

## Syntax

 **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 sublistvalue = sublist.getSublistValue({
    id : 'quantity',
    line: 1
})
...
//Add additional code
```

## Sublist.setSublistValue(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Sets the value of a sublist field.
---------------------------	------------------------------------

<b>Returns</b>	void
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.id	string	required	The internal ID name of the line item field being set.
options.line	number	required	The line number for this field.   <b>Note:</b> 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](#).

```
//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,
    label: 'Text Field'
});

sublist.setSublistValue({
    id : 'sublist',
    line : 2,
    value : "Text"
});
...

```

```
//Add additional code
```

## Sublist.updateTotallingFieldId(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Updates the ID of a field designated as a totalling column, which is used to calculate and display a running total for the sublist.
<b>Returns</b>	<a href="#">serverWidget.Sublist</a> object
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
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](#).

```
/**
 * @NApiVersion 2.x
 * @NScriptType suitelet
 */
define(["N/ui/serverWidget", "N/record"], function(serverWidget, record){
    return {
        onRequest: function (params)
        {
            var form = serverWidget.createForm({title: 'Simple Form'});
            var sublistObj2 = form.addSublist({id: 'mylist', type: serverWidget.SublistType.INLINEEDITOR, label: 'List'});
            sublistObj2.addField({id: 'description', type: serverWidget.FieldType.TEXT, label: 'Description'});
            sublistObj2.addField({id: 'amount', type: serverWidget.FieldType.CURRENCY, label: 'Amount'});
            sublistObj2.updateTotallingFieldId({id: 'amount'});
            sublistObj2.setSublistValue({id: 'description', line: 0, value: 'foo'});
            sublistObj2.setSublistValue({id: 'amount', line: 0, value: '10'});
            sublistObj2.setSublistValue({id: 'description', line: 1, value: 'bar'});
        }
    };
});
```

```

        sublistObj2.setSublistValue({id: 'amount', line: 1, value: '15'});
        form.addSublist({id: 'dummy', type: serverWidget.SublistType.STATICLIST, label: 'Dummy'});
        params.response.writePage( form );
    }
};

});

```

## Sublist.updateUniqueId(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Updates a field ID that is to have unique values across the rows in the sublist.
	 <b>Note:</b> This method is available on inlineeditor and editor sublists only.
<b>Returns</b>	serverWidget.Sublist object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
options.id	string	required	The internal ID name of the field to use as a unique 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](#).

```

//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.updateUniqueId({
    id : 'fieldid'
})
...
//Add additional code

```

## Sublist.displayType

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The display style for a sublist. Use the <a href="#">serverWidget.SublistDisplayType</a> enum to set this value.
<b>Type</b>	string
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	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 sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
sublist.displayType = serverWidget.SublistDisplayType.HIDDEN;
...
//Add additional code

```

## Sublist.helpText

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The inline help text for a sublist.
<b>Type</b>	string

<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
sublist.helpText = "Help Text Goes Here.";
...
//Add additional code
```

## Sublist.label

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The label for this sublist.
<b>Type</b>	string
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 sublist = form.addSublist({
```

```

        id : 'sublist',
        type : serverWidget.SublistType.INLINEEDITOR,
        label : 'Inline Editor Sublist'
});
var sublist = sublist.label;
...
//Add additional code

```

## Sublist.lineCount



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The number of line items on a sublist.
	<span style="border: 1px solid #0070C0; border-radius: 50%; padding: 2px 5px; font-size: small;">Info</span> <b>Note:</b> The first line number on a sublist is 0 (not 1).
<b>Type</b>	number (read-only)
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
var numLines = sublist.lineCount;
...
//Add additional code

```

## serverWidget.Tab



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates a tab or subtab on a <a href="#">serverWidget.Form</a> object. You can add a new tab or subtab to a form using one of the following methods:
---------------------------	--

	<ul style="list-style-type: none"> <li>▪ <a href="#">Form.addSubtab(options)</a></li> <li>▪ <a href="#">Form.addTab(options)</a></li> <li>▪ <a href="#">Form.insertSubtab(options)</a></li> <li>▪ <a href="#">Form.insertTab(options)</a></li> </ul> <p>The internal ID must be in lowercase, contain no spaces, and include the prefix <code>custpage</code> if you are adding the field to an existing page.</p> <div style="background-color: #e0f2ff; border: 1px solid #0070C0; padding: 10px; margin-top: 10px;"> <p><b>Note:</b> 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.</p> </div> <div style="background-color: #e0f2ff; border: 1px solid #0070C0; padding: 10px; margin-top: 10px;"> <p><b>Note:</b> 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.</p> </div>
	For a complete list of this object's properties, see <a href="#">Tab Object Members</a> .
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 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
```

## Tab.helpText



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The inline help text for a tab or subtab.
<b>Type</b>	string
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 tab = form.addTab({
    id : 'tabid',
    label : 'Tab'
});
tab.helpText = 'Help Text Goes Here';
...
//Add additional code
```

## Tab.label



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The label for a tab or subtab.
<b>Type</b>	string
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	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 tab = form.addTab({
    id : 'tabid',
    label : 'Tab'
});
var label = tab.label;
...
//Add additional code

```

## serverWidget.createAssistant(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates an assistant object.
<b>Returns</b>	serverWidget.Assistant object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.title	string	required	The title of the assistant. This title appears at the top of all assistant pages.	2015.2
options.hideNavBar	boolean <code>true</code>   <code>false</code>	optional	Indicates whether to hide the navigation bar menu.  By default, set to <code>false</code> . The header appears in the top-right corner on the assistant.  If set to <code>true</code> , the header on the assistant is hidden from view.	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'
});
...
//Add additional code

```

For more information, see the help topic [Sample Custom Assistant Script](#).

## serverWidget.createForm(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a form object.
<b>Returns</b>	<code>serverWidget.Form</code> object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.title</code>	string	required	The title of the form.	2015.2
<code>options.hideNavBar</code>	boolean <code>true</code>   <code>false</code>	optional	Indicates whether to hide the navigation bar menu.  By default, set to <code>false</code> . The header appears in the top-right corner on the form.  If set to <code>true</code> , the header on the assistant is hidden from view.	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'
}

```

```
});  
...  
//Add additional code
```

## serverWidget.createList(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Instantiates a standalone list.
<b>Returns</b>	<code>serverWidget.List</code> object
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.title</code>	string	required	The title of the list.	2016.1
<code>options.hideNavBar</code>	boolean <code>true</code>   <code>false</code>	optional	Indicates whether to hide the navigation bar menu.  By default, set to <code>false</code> . The header appears in the top-right corner on the form.  If set to <code>true</code> , the header on the assistant is hidden from view.	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/ui/serverWidget Module Script Samples](#).

```
//Add additional code  
...  
var list = serverWidget.createList({  
    title : 'Simple List'  
});  
...  
//Add additional code
```

## serverWidget.AssistantSubmitAction

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	<p>Holds the string values for submit actions performed by the user. This enum is used to set the value of the <a href="#">Assistant.getLastAction()</a>.</p> <p>After a <b>finish</b> action is submitted, by default, the text "Congratulations! You have completed the &lt;assistant title&gt;" appears on the finish page.</p> <p>In a non-sequential process (steps are unordered), <b>jump</b> is used to move to the user's last action.</p> <p> <b>Note:</b> 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>
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

## Values

Value
BACK
CANCEL
FINISH
JUMP
NEXT

## Syntax

 **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'
});
...
if (assistant.getLastAction() == serverWidget.AssistantSubmitAction.CANCEL) {
    ...
}
```

```
//Add additional code
```

## serverWidget.FieldBreakType



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for supported field break types. This enum is used to set the value of the <a href="#">Field.updateBreakType(options)</a> parameter when <a href="#">Field.updateBreakType(options)</a> is called.
	<p><b>Note:</b> 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>
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

## Values

Value	Description
NONE	This is the default value for field break type.
STARTCOL	This value moves the field into a new column. Additionally, it disables automatic field balancing if set on any field.
STARTROW	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 OUTSIDE BELOW.  For more information, see <a href="#">serverWidget.FieldLayoutType</a> and <a href="#">Field.updateLayoutType(options)</a> .

## Syntax



**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 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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for supported field display types. This enum is used to set the value of the <code>displayType</code> parameter when <code>Field.updateDisplayType(options)</code> is called.   <b>Note:</b> 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.
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

## Values

Value	Description of Field Type
DISABLED	Prevents a user from changing the field
ENTRY	The sublist field appears as a data entry input field (for a select field without a checkbox)
HIDDEN	The field on the form is hidden.
INLINE	The field appears as inline text
NORMAL	The field appears as a normal input field (for non-sublist fields)
READONLY	The field is disabled but it is still selectable and scrollable (for textarea fields)

## 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](#).

```

//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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	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.
	<p><b>Note:</b> 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>
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

## Values

Value	Description
STARTROW	This value makes the field appear first in a horizontally aligned field group in the normal field layout.
MIDROW	This value makes the field appear in the middle of a horizontally aligned field group in the normal field layout.
ENDROW	This value makes the field appear last in a horizontally aligned field group in the normal field layout.
OUTSIDE	This value makes the field appear outside (above or below based on form default) the normal field layout area.
OUTSIDE BELOW	This value makes the field appear below the normal field layout area. Using this allows you to position a field below a field group.
OUTSIDE ABOVE	This value makes the field appear above the normal field layout area. Using this allows you to position a field above a field group.
NORMAL	This value makes the fields appear in its default position.

## Syntax

**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 field = form.addField({
    id: 'custpage_text',
    type: serverWidget.FieldType.TEXT,
    label: 'Text'
});
field.updateLayoutType({
    layoutType: serverWidget.FieldLayoutType.OUTSIDE BELOW
});
...
//Add additional code
```

## serverWidget.FieldType

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	<p>Enumeration that holds the values for supported field types. This enum is used to set the value of the <code>type</code> parameter when <a href="#">Form.addField(options)</a> is called.</p> <p><b>Note:</b> 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><b>Important:</b> 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.</p>
<b>Supported Script Types</b>	<a href="#">SuiteScript 2.0 Suitelet Script Type</a> and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

## Values

<ul style="list-style-type: none"> <li>■ CHECKBOX</li> <li>■ CURRENCY</li> <li>■ DATE</li> <li>■ DATETIME</li> </ul>	<ul style="list-style-type: none"> <li>■ LONGTEXT</li> <li>■ MULTISELECT</li> <li>■ PASSWORD</li> <li>■ PERCENT</li> </ul>
--	--

▪ DATETIMETZ	▪ PHONE
▪ EMAIL	▪ SELECT
▪ FILE	▪ RADIO
▪ FLOAT	▪ RICHTEXT
▪ HELP	▪ TEXT
▪ INLINEHTML	▪ TEXTAREA
▪ INTEGER	▪ TIMEOFDAY
▪ IMAGE	▪ URL
▪ LABEL	

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 and RICHTEXT field types are not supported with [Sublist.addField\(options\)](#).
- The INLINEHTML field type should be considered as a 'write-only' type of field just used to add a field on a form.
- 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.
- Radio buttons that are inside one container are exclusive. The method **addField** on form has an optional parameter **container**. For an example, see [FieldGroup.label](#).

## Syntax



**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 field = form.addField({
    id : 'custpage_text',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
...
//Add additional code
```

## serverWidget.FormPageLinkType



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for supported page link types on a form. This enum is used to set the value of the <b>type</b> parameter when <a href="#">Form.addPageLink(options)</a> is called.
-------------------------	---

	<p><b>Note:</b> 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>
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

## Values

Value	Description
BREADCRUMB	Link appears on the top-left corner after the system bread crumbs
CROSSLINK	Link appears on the top-right corner

## Syntax

**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'
});
form.addPageLink({
    type : serverWidget.FormPageLinkType.CROSSLINK,
    title : 'NetSuite',
    url : 'http://www.netsuite.com'
})
...
//Add additional code
```

## serverWidget.LayoutJustification

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for supported justification layouts. This enum is used to set the value of the <code>align</code> parameter when <a href="#">List.addColumn(options)</a> is called.
	<p><b>Note:</b> 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>
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <code>beforeLoad(scriptContext)</code> )

<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

## Values

Value
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](#).

```
//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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for supported list styles. This enum is used to set the value of the <a href="#">List.style</a> property.
	<p><b>Note:</b> 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>
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <a href="#">beforeLoad(scriptContext)</a> )
<b>Module</b>	N/ui/serverWidget Module
<b>Since</b>	2015.2

## Values

Value
GRID
REPORT
PLAIN
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](#).

```
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});
list.style = serverWidget.ListStyle.REPORT;
...
//Add additional code
```

## serverWidget.SublistDisplayType

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for supported sublist display types. This enum is used to set the value of the <a href="#">Sublist.displayType</a> property.
	<p><b>Note:</b> 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>
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type ( <a href="#">beforeLoad(scriptContext)</a> )
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

## Values

Value
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](#).

```
//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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for valid sublist types. This enum is used to define the <code>type</code> parameter when <a href="#">Form.addSublist(options)</a> is called
	<p><b>Note:</b> 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>
<b>Supported Script Types</b>	SuiteScript 2.0 Suitelet Script Type and <a href="#">SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</a>
<b>Module</b>	<a href="#">N/ui/serverWidget Module</a>
<b>Since</b>	2015.2

## Values

Value	Description
INLINEEDITOR	These types of sublists are both fully editable. The only difference between these types is their appearance in the UI:
EDITOR	<ul style="list-style-type: none"> <li>■ 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.</li> <li>■ 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.</li> </ul>
LIST	This type of sublist has a fixed number of lines. You can update an existing line, but you cannot add lines to it.

Value	Description
	 <b>Note:</b> To make a field within a LIST type sublist editable, use <code>Field.updateDisplayType(options)</code> and the enum <code>serverWidget.FieldDisplayType</code> to update the field display type to ENTRY.
STATICLIST	This type of sublist is read-only. It cannot be edited in the UI, and it is not available for scripting.

## Syntax

 **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 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.

```
//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

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

Use the N/url module to determine URL navigation paths within NetSuite and format URL strings.

- N/url Module Members
- N/url Module Script Samples

**i Note:** If you have any hard-coded references to external URLs for Suitelets with the forms.netsuite.com domain, you must update these references. As of 2020.1, any external URL references with the old format will result in broken links. For access or redirection from another script to a Suitelet, the best practice is to use [url.resolveScript\(options\)](#) to discover the URL instead of hard-coding the URL. For more information about NetSuite domains, see the help topic [Understanding NetSuite URLs](#).

## N/url Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">url.format(options)</a>	string	Server-side scripts	Converts (serializes) URL query parameters into a string.
	<a href="#">url.resolveDomain(options)</a>	string	Client and server-side scripts	Returns a domain name for a NetSuite account.
	<a href="#">url.resolveRecord(options)</a>	string	Client and server-side scripts	Returns an internal URL string to a NetSuite record.
	<a href="#">url.resolveScript(options)</a>	string	Server-side scripts	Returns an external or internal URL string to a script.
	<a href="#">url.resolveTaskLink(options)</a>	string	Server-side scripts	Returns an internal URL for a tasklink.
Enum	<a href="#">url.HostType</a>	enum	Server-side scripts	An enum used to populate the hostType parameter of the <a href="#">url.resolveDomain(options)</a> method.

## N/url Module Script Samples

The following script samples show how to use the N/url module.

**i Note:** This sample script uses the **require** function so that you can copy it into the SuiteScript Debugger and test it. You must use the **define** function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to retrieve the relative URL of a record. With the internal ID value used in this sample, the returned output would be `/app/accounting/transactions/salesord.nl?id=6&e=T&compid='`, followed by the NetSuite account ID.

**⚠ Important:** The value used in this sample for the recordId field is a placeholder. Before using this sample, replace the recordId field value with a valid value from your NetSuite account. If you run a script with an invalid value, an error may occur.

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

```
require(['N/url'], function(url) {
  var output = url.resolveRecord({
    recordType: 'salesorder',
    recordId: 6,
    isEditMode: true
  });
});
```

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to generate an absolute URL to a specific resource.

**Important:** The value used in this sample for the recordId field is a placeholder. Before using this sample, replace the recordId field value with a valid value from your NetSuite account. If you run a script with an invalid value, an error may occur.

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

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();
});
```

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to get the domain for calling a RESTlet.

**Important:** The value used in this sample for the accountId field is a placeholder. Before using this sample, replace the accountId field value with a valid value from your NetSuite account. If you run a script with an invalid value, an error may occur.

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

```

require(['N/url'], function(url) {
    function resolveDomainUrl() {
        var sCompId = 'MSTRWLF';
        var output = url.resolveDomain({
            hostType: url.HostType.RESTLET,
            accountId: sCompId
        });
    }
    resolveDomainUrl();
});

```

**i Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample shows how to create a URL and do send a secure HTTPS POST request to that URL with an empty body. The server's response is logged.

**⚠ Important:** The value used in this sample for the `scriptId` and `deploymentId` fields are placeholders. Before using this sample, replace the `scriptId` and `deploymentId` values with valid values from your NetSuite account. If you run a script with an invalid value, an error may occur.

```

/**
 * @NApiVersion 2.x
 */

require(['N/url', 'N/https'], function(url, https) {
    var script = 'customscript1';
    var deployment = 'customdeploy1';
    var parameters = "";
    try {
        var suiteletURL = url.resolveScript({
            scriptId: script,
            deploymentId: deployment
        });
        var response = https.post({
            url: suiteletURL,
            body: parameters
        });
        log.debug(response.body.toString());
    }
    catch(e) {
        log.error(e.toString());
    }
});

```

## url.format(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

### Method Description

Creates a serialized representation of an object containing query parameters.

	Use the returned value to build a URL query string.
<b>Returns</b>	URL as a string
<b>Supported Script Types</b>	All server-side scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/url Module</a>
<b>Since</b>	2015.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.domain	string	required	The domain name.	2015.1
options.params	Object	required	Additional URL parameters as name/value pairs.	2015.1

## 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/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.

```
//Add additional code
...
var output = url.format({
    domain: 'http://fruitland.com',
    params: {
        fruit: 'grape',
        seedless: true,
        variety: 'Concord Giant',
        PLU: 4272
    }
});
...
//Add additional code
```

## url.resolveDomain(options)

<b>Method Description</b>	Returns a domain name for a NetSuite account.
<b>Returns</b>	string
<b>Supported Script Types</b>	Client and server-side scripts

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/url Module
<b>Since</b>	2017.1

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.hostType	string	required	The type of domain name you want to retrieve. Set this value using the <a href="#">url.HostType</a> enum.	2017.1
options.accountId	string	optional	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.  You can find the account ID at Setup > Company > Company Information in the <b>Account ID</b> field.	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/url Module Script Samples](#).

```
//Add additional code
...
var output = url.resolveDomain({
    hostType: url.HostType.APPLICATION,
    accountId: '012345'
});
...
//Add additional code
```

## url.resolveRecord(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the URL string to a NetSuite record.
<b>Returns</b>	URL to a NetSuite record as a string
<b>Supported Script Types</b>	Client and server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/url Module

<b>Since</b>	2015.1
--------------	--------

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.recordType	string	required	The type of record. For example, 'transaction'.	2015.1
options.recordId	string	required	The internal ID of the target record instance.	2015.1
options.isEditMode	boolean true   false	required	If set to <b>true</b> , returns a URL for the record in Edit mode.  If set to <b>false</b> , returns a URL for the record in View mode  The default value is <b>View</b> .	2015.1
options.params	Object	optional	Object used to add parameters for a custom URL. For example, a query to a database or to a search engine	2015.1

## 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/url Module Script Samples](#).

```
//Add additional code
...
var output = url.resolveRecord({
    recordType: 'salesorder',
    recordId: 6,
    isEditMode: true
});
...
//Add additional code
```

## url.resolveScript(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns an external or internal URL string to a script.
<b>Returns</b>	The URL as a string
<b>Supported Script Types</b>	All server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None

<b>Module</b>	N/url Module
<b>Since</b>	2015.1

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.scriptId	string	required	The internal ID of the script. The ID must identify a RESTlet or a Suitelet.	2015.1
options.deploymentId	string	required	The internal ID of the deployment script	2015.1
options.params	Object	optional	The object containing name/value pairs to describe the query.	2015.1
options.returnExternalUrl	boolean <code>true</code>   <code>false</code>	optional	Indicates whether to return the external URL.  By default, the internal URL is returned (i.e., the default value is false).	2015.1

## Syntax

 <b>Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/url Module Script Samples</a> .
---

```
//Add additional code
...
var output = url.resolveScript({
   scriptId: 'custom_script',
   deploymentId: 'custom_script_deployment',
    returnExternalUrl: true
});
...
//Add additional code
```

## url.resolveTaskLink(options)

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	Returns the internal URL to a NetSuite Tasklink.
<b>Returns</b>	The URL as a string
<b>Supported Script Types</b>	All server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None

<b>Module</b>	N/url Module
<b>Since</b>	2015.2

## Parameters

<b>i Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	Internal ID for the tasklink.  <b>i Note:</b> 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="nIPopupHelp('LIST_SCRIPT','help').	2015.1
options.params	Map	optional	The Map object containing name/value pairs to describe the query.	2015.1

## Syntax

<b>⚠ Important:</b> The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see <a href="#">N/url Module Script Samples</a> .
---

```
//Add additional code
...
u = url.resolveTaskLink('SRCH_JOB', p);
...
//Add additional code
```

## url.HostType

<b>Enum Description</b>	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 <a href="#">url.resolveDomain(options)</a> method.  <b>i Note:</b> 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.
<b>Type</b>	enum
<b>Supported Script Types</b>	All server-side scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/url Module</a>
<b>Since</b>	2017.1

## Values

Value	Description	Sample Result
APPLICATION	The domain for UI access.	<accountID>.app.netsuite.com <accountID> is replaced with your NetSuite account number.
FORM	The domain for forms hosted online, usually in Suitelets.	<accountID>.extforms.netsuite.com <accountID> is replaced with your NetSuite account number.
RESTLET	The domain for calling a RESTlet from an external source.	<accountID>.restlets.api.netsuite.com <accountID> is replaced with your NetSuite account number.
SUITETALK	The domain for SOAP web services requests.	<accountID>.suitetalk.api.netsuite.com <accountID> is replaced with your NetSuite account number.

For more information about NetSuite domains, see the help topic [Understanding NetSuite URLs](#).

 **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 [NetSuite Accounts Are Hosted in the Cloud](#).

## Syntax

 **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](#).

```
//Add additional code
...
var output = url.resolveDomain({
    hostType: url.HostType.APPLICATION,
    accountId: '012345'
});
...
//Add additional code
```

## N/util Module

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

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.



**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 Samples](#)

## N/util Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">util.isArray(obj)</a>	boolean true   false	Client and server-side scripts	Returns true if the <code>obj</code> parameter is a JavaScript and false otherwise.
	<a href="#">util.isBoolean(obj)</a>	boolean true   false	Client and server-side scripts	Returns true if the <code>obj</code> parameter is a Boolean and false otherwise.
	<a href="#">util.isDate(obj)</a>	boolean true   false	Client and server-side scripts	Returns true if the <code>obj</code> parameter is a JavaScript Date object and false otherwise.
	<a href="#">utilisFunction(obj)</a>	boolean true   false	Client and server-side scripts	Returns true if the <code>obj</code> parameter is a JavaScript Function object and false otherwise.
	<a href="#">util.isNumber(obj)</a>	boolean true   false	Client and server-side scripts	Returns true if the <code>obj</code> parameter is a JavaScript Number object or a value that evaluates to a Number object, and false otherwise.
	<a href="#">utilisObject(obj)</a>	boolean true   false	Client and server-side scripts	Returns true if the <code>obj</code> parameter is a strictly a JavaScript Object, and false otherwise.
	<a href="#">util.isRegExp(obj)</a>	boolean true   false	Client and server-side scripts	Returns true if the <code>obj</code> parameter is a JavaScript RegExp object or a value that evaluates to a RegExp object, and false otherwise.
	<a href="#">util.isString(obj)</a>	boolean true   false	Client and server-side scripts	Returns true if the <code>obj</code> parameter is a JavaScript String object or a value that evaluates to a String object, and false otherwise.
	<a href="#">util.nanoTime()</a>	number	Server-side scripts	Returns the amount of time elapsed from an arbitrary fixed point, in nanoseconds.
	<a href="#">util.each(iterable, callback)</a>	Object or Array	Client and server-side scripts	Iterates over each member in an Object or Array.
	<a href="#">util.extend(receiver, contributor)</a>	Object	Client and server-side scripts	Copies the properties in a source object to a destination object.

## N/util Module Script Samples

The following script samples demonstrate how to use the features of the N/util module.

## Sample 1: Iterate through an object

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample creates a sales order record. It uses the `util.each(iterable, callback)` method to set record fields based on the values in an iterable object. Make sure to replace hard-coded values (such as record IDs) with valid values from your NetSuite account.

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

require(['N/record'], function(record){
    // 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){      // (5, 39) and (1, 38)
        rec.selectNewLine('item');
        rec.setCurrentSublistValue('item', 'item', itemId);
        rec.setCurrentSublistValue('item', 'quantity', quantity);
        rec.commitLine('item');
    });

    var id = rec.save();
});
```

## util.isArray(obj)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns true if the <code>obj</code> parameter is a JavaScript Array object and false otherwise.
<b>Returns</b>	boolean true   false
<b>Supported Script Types</b>	All script types

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/util Module</a>
<b>Global object</b>	<a href="#">util Object</a>
<b>Since</b>	2016.1

## Parameters

Parameter	Type	Required / Optional	Description	Since
obj	Object   Primitive	Required	Object for which you want to verify the type.	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](#).

```
//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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns true if the <code>obj</code> parameter is a boolean and false otherwise.
<b>Returns</b>	boolean true   false
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/util Module</a>
<b>Since</b>	2016.1

## Parameters

Parameter	Type	Required / Optional	Description	Since
obj	Object   Primitive	Required	Object for which you want to verify the type.	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.

```
//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)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns true if the <code>obj</code> parameter is a JavaScript Date object and false otherwise.
<b>Returns</b>	boolean true   false
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/util Module</a>
<b>Since</b>	2016.1

## Parameters

Parameter	Type	Required / Optional	Description	Since
<code>obj</code>	Object   Primitive	Required	Object for which you want to verify the type.	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.

```
//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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns true if the <code>obj</code> parameter is a JavaScript Function object and false otherwise.
<b>Returns</b>	boolean true   false
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/util Module</a>
<b>Since</b>	2016.1

### Parameters

Parameter	Type	Required / Optional	Description	Since
<code>obj</code>	Object   Primitive	Required	Object for which you want to verify the type.	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](#).

```
//Add additional code
...
function test() {};
var test2 = function() {};

util.isFunction(test);    // returns true
util.isFunction(test2);  // returns true
...
//Add additional code
```

## util.isNumber(obj)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns true if the <code>obj</code> parameter is a JavaScript Number object or primitive, and false otherwise.
---------------------------	---

<b>Returns</b>	boolean true   false
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/util Module</a>
<b>Since</b>	2016.1

## Parameters

Parameter	Type	Required / Optional	Description	Since
obj	Object   Primitive	Required	Object for which you want to verify the type.	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](#).

```
//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
```

## utilisObject(obj)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns true if the <code>obj</code> parameter is a plain JavaScript object( <code>new Object()</code> or <code>{}</code> for example), and false otherwise.  Use this method, for example, to verify that a variable is a JavaScript object and not a JavaScript Function.
<b>Returns</b>	boolean true   false
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/util Module</a>
<b>Since</b>	2016.1

## Parameters

Parameter	Type	Required / Optional	Description	Since
obj	Object   Primitive	Required	Object for which you want to verify the type.	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.

```
//Add additional code
...
util.isObject({});           // returns true
util.isObject(function() {}); // returns false
...
//Add additional code
```

## util.isRegExp(obj)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns true if the <code>obj</code> parameter is a JavaScript RegExp object, and false otherwise.
<b>Returns</b>	boolean true   false
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/util Module</a>
<b>Since</b>	2016.1

## Parameters

Parameter	Type	Required / Optional	Description	Since
obj	Object   Primitive	Required	Object for which you want to verify the type.	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.

```
//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)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns true if the <code>obj</code> parameter is a JavaScript String object or primitive, and false otherwise
<b>Returns</b>	boolean true   false
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/util Module</a>
<b>Since</b>	2016.1

### Parameters

Parameter	Type	Required / Optional	Description	Since
<code>obj</code>	Object   Primitive	Required	Object for which you want to verify the type.	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](#).

```
//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()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the current time (epoch) in nanoseconds.  You can use this method to measure elapsed time between two events.
<b>Returns</b>	string
<b>Supported Script Types</b>	Server-side scripts

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/util Module</a>
<b>Since</b>	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](#).

```
//Add additional code
var startTime = util.nanoTime();
...
var elapsedTime = util.nanoTime() - startTime;
...
//Add additional code
```

## util.each(iterable, callback)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Iterates over each member in an Object or Array. This method calls the <code>callback</code> function on each member of the <code>iterable</code> .
<b>Returns</b>	The original collection as an Object   Array
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/util Module</a>
<b>Since</b>	2016.1

## Parameters

Parameter	Type	Required / Optional	Description	Since
iterable	Object   Array	Required	The data collection to iterate on	2016.1
callback	Function	Required	Takes the custom logic that you want to execute on each member of your collection of data.	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](#).

```
//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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	The Object receiving the properties copied from the contributor
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/util Module</a>
<b>Since</b>	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:

```

//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

```

```

    // Colors will get {'color1':'red','color2':'yellow','color3':'blue','color4':'green','color5':'orange',
    'color6':'violet'}
    util.extend(colors, secondSet);
});
...
//Add additional code

```

The following snippet shows overriding two objects with a few similar keys:

```

//Add additional code
...
var colors = {};
var firstSet = {'color1':'red',
    'color2':'yellow',
    'color3':'blue'
};
var secondSet = {'color2':'green',
    'color3':'orange',
    'color4':'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);

var x = 0;
});
...
//Add additional code

```

## N/workflow Module



**Note:** The content in this help topic pertains to SuiteScript 2.0.

Load the N/workflow module when you want to initiate new workflow instances or trigger existing workflow instances.

- [N/workflow Module Members](#)
- [N/workflow Module Script Sample](#)

### N/workflow Module Members

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">workflow.initiate(options)</a>	number	Server scripts	Initiates a workflow on-demand. This method is the programmatic equivalent

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
				<p>of the <a href="#">Initiate Workflow Action</a> action in SuiteFlow.</p> <p>Returns the internal ID of the workflow instance used to track the workflow against the record.</p>
	workflow.trigger(options)	number	Server scripts	<p>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.</p> <p>Returns the internal ID of the workflow instance used to track the workflow against the record.</p>

## N/workflow Module Script Sample

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample searches for a specific workflow deployed on the customer record and then executes it.

**Important:** This script sample uses placeholder values for the customer recordId and workflowId. Before using this sample, replace these IDs with valid values from your NetSuite account. If you run a script with an invalid value, the system may throw an error.

```
/*
 * @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
            });
            initiateWorkflow();
        }
    }
);
```

## workflow.initiate(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Initiates a workflow on-demand. This method is the programmatic equivalent of the <a href="#">Initiate Workflow Action</a> action in SuiteFlow.  Returns the internal ID of the workflow instance used to track the workflow against the record.  To asynchronously initiate a workflow, see <a href="#">task.WorkflowTriggerTask</a> .
<b>Returns</b>	number
<b>Supported Script Types</b>	All server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>
<b>Governance</b>	20 usage units
<b>Module</b>	<a href="#">N/workflow Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.recordType	string	required	The record type ID of the workflow base record. Use values from the <a href="#">record.Type</a> enum. This is the <b>Record Type</b> field on the <a href="#">Workflow Definition Page</a> .	2015.2
options.recordId	string   number	required	The internal ID of the base record.	2015.2
options.workflowId	string   number	required	The internal ID (number) or script ID (string) for the workflow definition. This is the <b>ID</b> field on the <a href="#">Workflow Definition Page</a> .	2015.2
options.defaultValues	Object	optional	The object that contains key/value pairs to set default values on fields specific to the workflow.  These can include fields on the <a href="#">Workflow Definition Page</a> or workflow and state <a href="#">Workflow Custom Fields</a> .	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/workflow Module Script Sample](#).

```
//Add additional code
...
var workflowInstanceId = workflow.initiate({
    recordType: 'customer',
```

```

    recordId: 24,
    workflowId: 'customworkflow_myWorkFlow'
});
...
//Add additional code

```

## workflow.trigger(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	number
<b>Supported Script Types</b>	All server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>
<b>Governance</b>	20 usage units
<b>Module</b>	N/workflow Module
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.recordType	string	required	The record type ID of the workflow base record. Use values from the <a href="#">record.Type</a> enum. This is the <b>Record Type</b> field on the <a href="#">Workflow Definition Page</a> .	2015.2
options.recordId	string   number	required	The internal ID of the base record.	2015.2
options.workflowId	string   number	required	The internal ID (number) or script ID (string) for the workflow definition. This is the <b>ID</b> field on the <a href="#">Workflow Definition Page</a> .	2015.2
options.workflowInstanceId	string   number	optional	The internal ID of the workflow instance.	2015.2
options.actionId	string   number	optional	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.	2015.2
options.stateId	string   number	optional	The internal ID (number) or script ID (string) of the workflow instance.	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/workflow Module Script Sample](#).

```
//Add additional code
...
var workflowInstanceId = workflow.trigger({
    recordType: 'salesorder',
    recordId: 1234,
    workflowId: 'custworkflow_name',
    defaultValues: p
    actionId: workflowaction25
});
...
//Add additional code
```

## N/xml Module

**Note:** The content in this help topic pertains to SuiteScript 2.0.

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

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Object	<a href="#">xml.Parser</a>	Object	Client and server-side scripts	Encapsulates the functionality used by NetSuite to parse XML.
	<a href="#">xml.XPath</a>	Object	Client and server-side scripts	Encapsulates the functionality used by NetSuite to run XPath expressions. XPath is a standard for enumerating paths in an XML document collection.
	<a href="#">xml.Node</a>	Object	Client and server-side scripts	Represents a generic XML node in an XML document. A node can be a Document, Element, or Attribute.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	<a href="#">xml.Document</a>	Object	Client and server-side scripts	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 <a href="#">xml.Document</a> object to manipulate the XML document and the nodes in the document tree.
	<a href="#">xml.Element</a>	Object	Client and server-side scripts	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.
	<a href="#">xml.Attr</a>	Object	Client and server-side scripts	Represents an attribute node of an <a href="#">xml.Element</a> object.
Method	<a href="#">xml.escape(options)</a>	string	Client and server-side scripts	Prepares a string for use in XML by escaping XML markup, such as angle brackets, quotation marks, and ampersands.
	<a href="#">xml.validate(options)</a>	void	Server-side scripts	Validates an XML document against an XML Schema (XSD).
Enum	<a href="#">xml.NodeType</a>	string (read-only)	Client and server-side scripts	Enumeration that holds the string values for the supported node types. The <a href="#">Node.nodeType</a> property is defined by one of the values in this enum.

## Parser Object Members

The following members are called on the [xml.Parser](#) object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">Parser.fromString(options)</a>	<a href="#">xml.Document</a>	Client and server-side scripts	Parses a string into a W3C XML document object.
	<a href="#">Parser.toString(options)</a>	string	Client and server-side scripts	Converts (serializes) an <a href="#">xml.Document</a> object into a string.

## XPath Object Members

The following members are called on the [xml.XPath](#) object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">XPath.select(options)</a>	<a href="#">xml.Node[]</a>	Client and server-side scripts	Selects an array of nodes from an XML document using an XPath expression.

## Node Object Members

The following members are called on the [xml.Node](#) object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">Node.appendChild(options)</a>	xml.Node	Client and server-side scripts	Appends a node after the last child node of a specific element node. Returns the new child node.
	<a href="#">Node.cloneNode(options)</a>	xml.Node	Client and server-side scripts	Creates a copy of a node. Returns the copied node.
	<a href="#">Node.compareDocumentPosition(options)</a>	number	Client and server-side scripts	Returns a number that reflects where two nodes are located, compared to each other.
	<a href="#">Node.hasAttributes()</a>	boolean true   false	Client and server-side scripts	Returns <b>true</b> if the current node has child nodes or returns <b>false</b> if the current node does not have child nodes.
	<a href="#">Node.hasChildNodes()</a>	boolean true   false	Client and server-side scripts	Returns <b>true</b> if the current node has any attributes. Note that only element nodes can have attributes.
	<a href="#">Node.insertBefore(options)</a>	xml.Node	Client and server-side scripts	Inserts a new child node before an existing child node for the current node.
	<a href="#">Node.isDefaultNamespace(options)</a>	boolean true   false	Client and server-side scripts	Returns <b>true</b> if the specified namespace uniform resource identifier (URI) is the default namespace for the current node or returns <b>false</b> if the specified namespace is not the default namespace.
	<a href="#">Node.isEqualNode(options)</a>	boolean true   false	Client and server-side scripts	Returns <b>true</b> if two nodes are equal or returns <b>false</b> if two nodes are not equal.
	<a href="#">Node.isSameNode(options)</a>	boolean true   false	Client and server-side scripts	Returns <b>true</b> if two nodes reference the same object or returns <b>false</b> if two nodes do not reference the same object.
	<a href="#">Node.lookupNamespaceURI(options)</a>	string	Client and server-side scripts	Returns the namespace uniform resource identifier (URI) that matches the specified namespace prefix.
	<a href="#">Node.lookupPrefix(options)</a>	string	Client and server-side scripts	Returns the namespace prefix associated with the specified namespace uniform resource identifier (URI).
	<a href="#">Node.normalize()</a>	void	Client and server-side scripts	Puts all text nodes underneath a node, including attribute nodes, into a normal form.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	Node.removeChild(options)	xml.Node	Client and server-side scripts	Removes the specified child node. Returns the removed child node.
	Node.replaceChild(options)	xml.Node	Client and server-side scripts	Replaces a specific child node with another child node in a list of child nodes.
Property	Node.attributes	Object (read-only)	Client and server-side scripts	Key-value pairs for all attributes for an <a href="#">xml.Element</a> node. Returns <b>null</b> for all other node types.
	Node.baseURI	string (read-only)	Client and server-side scripts	Absolute base uniform resource identifier (URI) of a node or <b>null</b> if the URI cannot be determined.
	Node.childNodes	xml.Node[] (read-only)	Client and server-side scripts	Array of all child nodes of a node or an empty array if there are no child nodes.
	Node.firstChild	xml.Node (read-only)	Client and server-side scripts	First child node for a specific node or <b>null</b> if there are no child nodes.
	Node.lastChild	xml.Node (read-only)	Client and server-side scripts	Last child node for a specific node or <b>null</b> if there is no last child node.
	Node.localName	string (read-only)	Client and server-side scripts	The local part of the qualified name of a node.
	Node.namespaceURI	string (read-only)	Client and server-side scripts	The namespace uniform resource identifier (URI) of a node or <b>null</b> if there is no namespace URI for the node.
	Node.nextSibling	xml.Node (read-only)	Client and server-side scripts	The next node in a node list or <b>null</b> if the current node is the last node.
	Node.nodeName	string (read-only)	Client and server-side scripts	Name of a node, depending on the type. For example, for a node of type <a href="#">xml.Element</a> , the name is the name of the element.
	Node.nodeType	string	Client and server-side scripts	The type of node defined as a value from the <a href="#">xml.NodeType</a> enum.
	Node.nodeValue	string	Client and server-side scripts	The value of a node, depending on its type.
	Node.ownerDocument	xml.Document (read-only)	Client and server-side scripts	The root element for a node as a <a href="#">xml.Document</a> object.
	Node.parentNode	xml.Node (read-only)	Client and server-side scripts	The parent node of a node.
	Node.prefix	string	Client and server-side scripts	The namespace prefix of the node, or <b>null</b> if the node does not have a namespace.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	Node.previousSibling	xml.Node (read-only)	Client and server-side scripts	The previous node in a node list or <b>null</b> if the current node is the first node.
	Node.textContent	string	Client and server-side scripts	The textual content of a node and its descendants.

## 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.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	Document.adoptNode(options)	xml.Node	Client and server-side scripts	Attempts to adopt a node from another document to this document.
	Document.createAttribute(options)	xml.Attr	Client and server-side scripts	Creates an attribute node of type ATTRIBUTE_NODE with the optional specified value.
	Document.createAttributeNS(options)	xml.Attr	Client and server-side scripts	Creates an attribute node of type ATTRIBUTE_NODE, with the specified namespace value and optional specified value.
	Document.createCDATASection(options)	xml.Node	Client and server-side scripts	Creates a CDATA section node of type DOCUMENT_FRAGMENT_NODE with the specified data.
	Document.createComment(options)	xml.Node	Client and server-side scripts	Creates a Comment node of type COMMENT_NODE with the specified string.
	Document.createDocumentFragment()	xml.Node	Client and server-side scripts	Creates a node of type DOCUMENT_FRAGMENT_NODE.
	Document.createElement(options)	xml.Element	Client and server-side scripts	Creates a new node of type ELEMENT_NODE with the specified name.
	Document.createElementNS(options)	xml.Element	Client and server-side scripts	Creates a new node of type ELEMENT_NODE with the specified namespace URI and name.
	Document.createProcessingInstruction(options)	xml.Node	Client and server-side scripts	Creates a new node of type PROCESSING_INSTRUCTION_NODE with the specified target and data.
	Document.createTextNode(options)	xml.Node	Client and server-side scripts	Creates a new node of type TEXT_NODE.
	Document.getElementById(options)	xml.Element	Client and server-side scripts	Returns the element that has an ID attribute with the specified value as an <code>xml.Element</code> object.
	Document.getElementsByTagName(options)	xml.Element[]	Client and server-side scripts	Returns an array of <code>xml.Element</code> objects with a specific tag name, in

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
				the order in which they appear in the XML document.
	<a href="#">Document.getElementsByTagNameNS(options)</a>	<code>xml.Element[]</code>	Client and server-side scripts	Returns an array of <code>xml.Element</code> objects with a specific tag name and namespace, in the order in which they appear in the XML document.
	<a href="#">Document.importNode(options)</a>	<code>xml.Node</code>	Client and server-side scripts	Imports a node from another document to this document. Creates a new copy of the source node.
Property	<a href="#">Document.doctype</a>	Object (read-only)	Client and server-side scripts	Returns a node of type <code>DOCUMENT_TYPE_NODE</code> that represents the doctype of the XML document.
	<a href="#">Document.documentElement</a>	<code>xml.Element</code> (read-only)	Client and server-side scripts	Root node of the XML document.
	<a href="#">Document.documentElementURI</a>	string (read-only)	Client and server-side scripts	Location of the document or <code>null</code> if undefined.
	<a href="#">Document.inputEncoding</a>	string (read-only)	Client and server-side scripts	Encoding used for an XML document at the time the document was parsed.
	<a href="#">Document.xmlEncoding</a>	string (read-only)	Client and server-side scripts	Part of the XML declaration, the XML encoding of the XML document.
	<a href="#">Document.xmlStandalone</a>	boolean <code>true</code>   <code>false</code>	Client and server-side scripts	Part of the XML declaration, returns <code>true</code> if the current XML document is standalone or returns <code>false</code> if it is not.
	<a href="#">Document.xmlVersion</a>	string	Client and server-side scripts	Part of the XML declaration, the version number of the XML document.

## 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.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Method	<a href="#">Element.getAttribute(options)</a>	string	Client and server-side scripts	Returns the value of the specified attribute.
	<a href="#">Element.getAttributeNode(options)</a>	<code>xml.Attr</code>	Client and server-side scripts	Retrieves an attribute node by name.
	<a href="#">Element.getAttributeNodeNS(options)</a>	string	Client and server-side scripts	Returns an attribute node with the specified namespace URI and local name.
	<a href="#">Element.getAttributeNS(options)</a>	<code>xml.Attr</code>	Client and server-side scripts	Returns an attribute value with the specified namespace URI and local name.
	<a href="#">Element.getElementsByTagNameByTagName(options)</a>	<code>xml.Element[]</code>	Client and server-side scripts	Returns an array of descendant <code>xml.Element</code> objects with a specific tag

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
				name, in the order in which they appear in the XML document.
	Element.getElementsByTagNameNS(options)	xml.Element[]	Client and server-side scripts	Returns an array of descendant <code>xml.Element</code> objects with a specific tag name and namespace, in the order in which they appear in the XML document.
	Element.hasAttribute(options)	boolean <code>true</code>   <code>false</code>	Client and server-side scripts	Returns <code>true</code> if the current element has an attribute with the specified name or if that attribute has a default value. Otherwise, returns <code>false</code> .
	Element.hasAttributeNS(options)	boolean <code>true</code>   <code>false</code>	Client and server-side scripts	Returns <code>true</code> if the current element has an attribute with the specified local name and namespace or if that attribute has a default value. Otherwise, returns <code>false</code> .
	Element.removeAttribute(options)	void	Client and server-side scripts	Removes the attribute with the specified name.
	Element.removeAttributeNode(options)	xml.Attr	Client and server-side scripts	Removes the attribute specified as a <code>xml.Attr</code> object.
	Element.removeAttributeNS(options)	void	Client and server-side scripts	Removes the attribute with the specified namespace URI and local name.
	Element.setAttribute(options)	void	Client and server-side scripts	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.
	Element.setAttributeNode(options)	xml.Attr	Client and server-side scripts	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.
	Element.setAttributeNodeNS(options)	xml.Attr	Client and server-side scripts	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.
	Element.setAttributeNS(options)	void	Client and server-side scripts	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.
Property	Element.tagName	string (read-only)	Client and server-side scripts	The tag name of this <code>xml.Element</code> object.

## Attr Object Members

The following members are called on the `xml.Attr` object.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
Property	Attr.name	string (read-only)	Client and server-side scripts	The name of an attribute.

Member Type	Name	Return Type / Value Type	Supported Script Types	Description
	Attr.ownerElement	xml.Element (read-only)	Client and server-side scripts	The <a href="#">xml.Element</a> object that is the parent of the <a href="#">xml.Attr</a> object.
	Attr.specified	boolean <b>true</b>   <b>false</b>	Client and server-side scripts	Returns <b>true</b> if the attribute value is set in the parsed XML document, and <b>false</b> if it is a default value in a DTD or Schema.
	Attr.value	string	Client and server-side scripts	Value of an attribute. The value of the attribute is returned as a string. Character and general entity references are replaced with their values.

## N/xml Module Script Samples

The following script samples demonstrate how to use the features of the N/xml module. These samples reference the following XML file called BookSample.xml:

```
<bookstore xmlns:b="http://www.qualifiednamespace.com/book">
  <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:year>2003</b:year>
    <b:price>39.95</b:price>
  </b:book>
</bookstore>
```

## Sample 1: Load an XML file and obtain child element values



**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample loads the BookSample.xml file from the File Cabinet, iterates through the individual book nodes, and accesses the child node values.

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

require(['N/xml', 'N/file'], function(xml, file) {
    return {
        onRequest: function(options) {
            var sentence = '';
            var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();

            var xmlDoc = xml.Parser.fromString({
                text: xmlFileContent
            });
            var bookNode = xmlDoc.XPath.select({
                node: xmlDoc,
                xpath: '//book'
            });

            for (var i = 0; i < bookNode.length; i++) {
                var title = bookNode[i].nextSibling.textContent;
                var author = bookNode[i].getElementsByTagName({
                    tagName: 'b:author'
                })[0].textContent;
                sentence += 'Author: ' + author + ' wrote ' + title + '.\n';
            }

            options.response.write(sentence);
        }
    };
});
```

This script produces the following output when used with the BookSample.xml file:

```
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.
```

## Sample 2: Parse an XML file and log element values

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following sample parses the XML string stored in the `xmlString` variable. The sample selects all `config` elements in the `xmlDocument` node, loops through them, and logs their contents.

```
/*
 * @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);
    }
});
```

## Sample 3: Modify an XML file

**Note:** This sample script uses the `require` function so that you can copy it into the SuiteScript Debugger and test it. You must use the `define` function in an entry point script (the script you attach to a script record and deploy). For more information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

The following samples shows how to modify an XML file.

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

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



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the functionality used by NetSuite to parse an XML document. For a complete list of this object's methods, see <a href="#">Parser Object Members</a> .
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Syntax

```

//Add additional code
...
var parserObj = xml.Parser;
...
//Add additional code

```

## Parser.fromString(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Parses 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. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <b>Note:</b> You can also use this method to validate your XML. If you pass a malformed string in as the <code>options.text</code> argument, <code>Parser.fromString</code> returns an <code>SSS_XML_DOM_EXCEPTION</code> error.         </div>
<b>Returns</b>	<code>xml.Document</code>

<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.text	string	Required	String being converted to an <a href="#">xml.Document</a> .

## Errors

Error Code	Thrown If
SSS_XML_DOM_EXCEPTION	The input XML string is malformed.

## Syntax

```
//Add additional code
...
var xmlDocument = xml.Parser.fromString({
    text : xmlStringContent
});
...
//Add additional code
```

## Parser.toString(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Converts (serializes) an <a href="#">xml.Document</a> object into a string. This API is useful, for example, if you want to serialize and store an <a href="#">xml.Document</a> in a custom field.
<b>Returns</b>	string
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.document	<a href="#">xml.Document</a>	Required	XML document being serialized.

## Syntax

```
//Add additional code
...
var xmlStringContent = xml.Parser.toString({
    document : xmlDocument
});
...
//Add additional code
```

## xml.XPath

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Encapsulates the functionality to run XPath expressions. For a complete list of this object's methods, see <a href="#">XPath Object Members</a> .
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var xpath = xml.XPath;
...
//Add additional code
```

## XPath.select(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Selects an array of nodes from an XML that match an XPath expression.
<b>Returns</b>	<a href="#">xml.Node[]</a>

<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.node	<a href="#">xml.Node</a>	Required	XML node being queried.
options.xpath	string	Required	XPath expression used to query node.

## Syntax

```
//Add additional code
...
var bookNode = xml.XPath.select({
    node : xmlDocument,
    xpath : '//book'
});
...
//Add additional code
```

## xml.Node

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Represents a single node in an XML document tree. The XML DOM presents a document as a hierarchy of node objects. See the <a href="#">xml.NodeType</a> 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 <a href="#">Node Object Members</a> .  For other code snippets that use this object, see the syntax sample that follows, as well as <a href="#">Node.childNodes</a> and N/xml Module Script Sample.
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var bookNode = xml.XPath.select({
  node : xmlDocument,
  xpath : '//book'
});
...
//Add additional code
```

## Node.appendChild(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Appends a node after the last child node of a specific element node. Returns the new child node.
<b>Returns</b>	<a href="#">xml.Node</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.newChild	<a href="#">xml.Node</a>	Required	<a href="#">xml.Node</a> object to append.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	HIERARCHY_REQUEST_ERR: An attempt was made to insert a node where it is not permitted.	Node cannot be appended.

## Syntax

```
//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)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a copy of a node. Returns the copied node.
<b>Returns</b>	<a href="#">xml.Node</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
options.deep	boolean <code>true</code>   <code>false</code>	Optional	Use <code>true</code> to clone the node, attributes, and all descendants. Use <code>false</code> to only clone the node and attributes.

### Syntax

```

//Add additional code
...
var copiednode = elem[0].cloneNode({
    deep : true
});
...
//Add additional code

```

## Node.compareDocumentPosition(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns a number that reflects where two nodes are located, compared to each other. Returns one of the following numbers: <ul style="list-style-type: none"><li>■ 1. The two nodes do not belong to the same document.</li><li>■ 2. The specified node comes before the current node.</li><li>■ 4. The specified node comes after the current node.</li><li>■ 8. The specified node contains the current node.</li><li>■ 16. The current node contains the specified node.</li><li>■ 32. The specified and current nodes do not have a common container node or the two nodes are different attributes of the same node.</li></ul>
	<p><b>Note:</b> 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.</p>
	<p><b>Important:</b> This method is not supported on Internet Explorer.</p>
<b>Returns</b>	number
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description
options.other	<a href="#">xml.Node</a>	Required	The node to compare with the current node.

### Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	Invalid argument type, expected xml.Node or subclass: other	The options.other is of type <a href="#">xml.Node</a> .

### Syntax

```
//Add additional code
```

```

...
var posCode = elem[0].compareDocumentPosition({
    other : parentNode[0]
});
...
//Add additional code

```

## Node.hasAttributes()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns <b>true</b> if the current node has attributes defined, or <b>false</b> otherwise.
	 <b>Important:</b> This method is not supported on Internet Explorer.
<b>Returns</b>	boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Syntax

```

//Add additional code
...
var hasAttributes = parentNode[0].hasAttributes()
...
//Add additional code

```

## Node.hasChildNodes()

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns <b>true</b> if the current node has child nodes or returns <b>false</b> if the current node does not have child nodes.
<b>Returns</b>	boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var hasChildren = parentNode[0].hasChildNodes()
...
//Add additional code
```

## Node.insertBefore(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	xml.Node
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.newChild	xml.Node	Required	The new child node to insert.
options.refChild	xml.Node	Required	The node before which to insert the new child node.  If <b>refChild</b> is , the method inserts the new node at the end of the list of children.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	HIERARCHY_REQUEST_ERR: An attempt was made to insert a node where it is not permitted.	Node cannot be inserted.

## Syntax

```
//Add additional code
...
var insertednode = parentNode[0].insertBefore({
    newChild : elemList1[0],
```

```

    refChild : elemList2[0]
});
...
//Add additional code

```

## Node.isDefaultNamespace(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.  See also <a href="#">Node.namespaceURI</a> .
<b>Returns</b>	boolean <code>true</code>   <code>false</code>
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.namespaceURI</code>	string	Required	The namespace URI to compare.

### Syntax

```

//Add additional code
...
var isDefault = parentNode[0].isDefaultNamespace({
    namespaceURI : '*'
});
...
//Add additional code

```

## Node.isEqualNode(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns <code>true</code> if two nodes are equal or returns <code>false</code> if two nodes are not equal.
---------------------------	--

	<p>The two nodes are equal if they meet the following conditions:</p> <ul style="list-style-type: none"> <li>■ Both nodes have the same type.</li> <li>■ Both nodes have the same attributes and attribute values. The order of the attributes is not considered.</li> <li>■ Both nodes have equal lists of child nodes and the child nodes appear in the same order.</li> </ul>
	<p> <b>Note:</b> Two nodes may be equal, even if they are not the same. See <a href="#">Node.isSameNode(options)</a>.</p>
	<p> <b>Important:</b> This method is not supported on Internet Explorer.</p>
<b>Returns</b>	boolean <code>true</code>   <code>false</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description
<code>options.other</code>	<a href="#">xml.Node</a>	Required	The node to compare with the current node.

## Syntax

```
//Add additional code
...
var isEqual = elem[0].isEqualNode({
    other : node
});
...
//Add additional code
```

## Node.isSameNode(options)

 <b>Note:</b>	The content in this help topic pertains to SuiteScript 2.0.
--	---

<b>Method Description</b>	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.  If two nodes are the same, all attributes have the same values and you can use methods on the two nodes interchangeably.
---------------------------	---

	<p><b>Note:</b> Two nodes that are the same are also equal. See <a href="#">Node.isEqualNode(options)</a>.</p>
	<p><b>Important:</b> This method is not supported on Internet Explorer or Firefox.</p>
<b>Returns</b>	boolean <code>true</code>   <code>false</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description
<code>options.other</code>	<a href="#">xml.Node</a>	Required	The node to compare with the current node.

## Syntax

```
//Add additional code
...
var isSame = elem[0].isSameNode({
    other : node
});
...
//Add additional code
```

## Node.lookupNamespaceURI(options)

<b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.
--

<b>Method Description</b>	Returns the namespace uniform resource identifier (URI) that matches the specified namespace prefix.  Returns <code>null</code> if the specified prefix does not have an associated URI.
<b>Returns</b>	<code>string</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None

<b>Module</b>	N/xml Module
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description
options.prefix	string	Required	Namespace prefix associated with the namespace URI.

## Syntax

```
//Add additional code
...
var uri = parentNode[0].lookupNamespaceURI({
    prefix : '*'
});
...
//Add additional code
```

## Node.lookupPrefix(options)

 <b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.
---

<b>Method Description</b>	Returns the namespace prefix associated with the specified namespace uniform resource identifier (URI).  Returns <b>null</b> 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.   <b>Important:</b> This method is not supported on Internet Explorer.
<b>Returns</b>	string
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/xml Module
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description
options.namespaceURI	string	Required	Namespace URI associated the namespace prefix.

## Syntax

```
//Add additional code
...
var prefix = parentNode[0].lookupPrefix({
    namespaceURI : '*'
});
...
//Add additional code
```

## Node.normalize()



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	void
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
node.normalize();
...
//Add additional code
```

## Node.removeChild(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Removes the specified child node.
<b>Returns</b>	<a href="#">xml.Node</a>
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Governance</b>	None
<b>Module</b>	N/xml Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.oldChild	xml.Node	Required	Node to remove.

## Errors

Error Code	Message	Thrown If
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

```
//Add additional code
...
var removednode = parentNode[0].removeChild({
    oldChild : node
});
...
//Add additional code
```

## Node.replaceChild(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	xml.Node
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/xml Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.newChild	<a href="#">xml.Node</a>	Required	New child node to add.
options.oldChild	<a href="#">xml.Node</a>	Required	Child node to replaced with the new node.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	NOT_FOUND_ERR: An attempt is made to reference a node in a context where it does not exist.	Child node cannot be found.
SSS_XML_DOM_EXCEPTION	HIERARCHY_REQUEST_ERR: An attempt was made to insert a node where it is not permitted.	Child node cannot be replaced.

## Syntax

```
//Add additional code
...
var replacednode = parentNode.replaceChild({
    newChild : elem[2],
    oldChild : elem[1]
});
...
//Add additional code
```

## Node.attributes

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Key-value pairs for all attributes for an <a href="#">xml.Element</a> node. Returns <code>null</code> for all other node types.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var attrs = elem[0].attributes;
...
```

```
//Add additional code
```

## Node.baseURI

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Absolute base uniform resource identifier (URI) of a node or <code>null</code> if the URI cannot be determined.  For client scripts, this property always returns <code>null</code> .
	<b>i Note:</b> The format of this value is browser-specific.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Syntax

```
//Add additional code
...
var baseuri = parentNode[0].baseURI;
...
//Add additional code
```

## Node.childNodes

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Array of all child nodes of a node or an empty array if there are no child nodes.
<b>Type</b>	<a href="#">xml.Node[]</a>
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Syntax

```
//Add additional code
...
var childnodes = parentNode[0].childNodes;
...
//Add additional code
```

## Node.firstChild

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The first child node of a node, or <code>null</code> if there are no child nodes.
<b>Type</b>	<code>xml.Node</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Syntax

```
//Add additional code
...
var nodeValue1 = bookNode[0].firstChild.nextSibling.textContent;
...
//Add additional code
```

## Node.lastChild

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The last child node of a node, or <code>null</code> if there are no child nodes.
<b>Type</b>	<code>xml.Node</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Syntax

```
//Add additional code
...
var nodeValue = parentNode[0].lastChild.previousSibling.textContent;
...
//Add additional code
```

## Node.localName

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The local part of the qualified name of a node.
-----------------------------	---

<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var localname = parentNode[0].localName;
...
//Add additional code
```

## Node.namespaceURI

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The namespace uniform resource identifier (URI) of a node or <b>null</b> if there is no namespace URI for the node.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var uri = parentNode[0].namespaceURI;
...
//Add additional code
```

## Node.nextSibling

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The next node in a node list or <b>null</b> if the current node is the last node.
<b>Type</b>	<a href="#">xml.Node</a> (read-only)
<b>Supported Script Types</b>	All script types

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var nodeName = parentNode[0].firstChild.nextSibling.textContent;
...
//Add additional code
```

## Node.nodeName

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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.
	 <b>Note:</b> On Chrome, this property also includes the namespace or prefix.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var nodeName = parentNode[0].firstChild.nodeName;
...
//Add additional code
```

## Node.nodeType

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The type of node as an enum. For all possible values of this property, see <a href="#">xml.NodeType</a> .
<b>Type</b>	<code>xml.NodeType</code>
<b>Supported Script Types</b>	All script types

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var nodeType = parentNode[0].firstChild.nodeType;
...
//Add additional code
```

## Node.nodeValue



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The value of a node, depending on its type. If the value is <code>null</code> , setting this value has no effect.
<b>Type</b>	<code>string</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var nodeValue = parentNode[0].firstChild.nodeValue;
...
//Add additional code
```

## Node.ownerDocument



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The root element for a node as a <code>xml.Document</code> object. Use this object to create new nodes with <code>Document.createElement(options)</code> or <code>Document.createElementNS(options)</code> .
<b>Type</b>	<code>xml.Document</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>

<b>Since</b>	2015.2
--------------	--------

## Syntax

```
//Add additional code
...
var doc = parentNode[0].ownerDocument;
...
//Add additional code
```

## Node.parentNode

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The parent node of a node. All node types, except <code>xml.Attr</code> , <code>xml.Document</code> , <code>DocumentFragment</code> , <code>Entity</code> , and <code>Notation</code> can have a parent node.  See <a href="#">xml.NodeType</a> for possible node types.
<b>Type</b>	<code>xml.Node</code>
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code ...
var nodevalue = parentNode[0].lastChild.parentNode.textContent;
...
//Add additional code
```

## Node.prefix

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The namespace prefix of the node, or <code>null</code> if the node does not have a namespace. If the value is <code>null</code> , setting it has no effect, including read-only node types.
<b>Type</b>	<code>string</code>
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	NAMESPACE_ERR: An attempt is made to create or change an object in a way which is incorrect with regard to namespaces.	Cannot edit the node prefix.

## Syntax

```
//Add additional code
...
var namespacePrefix = parentNode[0].firstChild.prefix;
...
//Add additional code
```

## Node.previousSibling

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The previous node in a node list or <code>null</code> if the current node is the first node.
<b>Type</b>	<code>xml.Node</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var nodeValue = parentNode[0].lastChild.previousSibling.textContent;
...
//Add additional code
```

## Node.textContent

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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.
<b>Type</b>	<code>string</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Module</b>	N/xml Module
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var nodeValue = parentNode[0].firstChild.textContent;
...
//Add additional code
```

## xml.Document



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	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 <a href="#">Document Object Members</a> .  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 <a href="#">Node Object Members</a> .
<b>Supported Script Types</b>	All script types
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/xml Module
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var xmlDoc = xml.Parser.fromString({
    text : xmlFileContent
});
...
//Add additional code
```

## Document.adoptNode(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
	<b>⚠ Important:</b> This method is not supported on Internet Explorer.
<b>Returns</b>	<a href="#">xml.Node</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.source	<a href="#">xml.Node</a>	Required	Source node to add as a child into the current node object.

## Errors

Error Code	Message	Thrown If
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 adopted.

## Syntax

```
//Add additional code
...
var adoptedNode = xmlDocument1.adoptNode({
    source : sourceNode,
});
...
//Add additional code
```

## Document.createAttribute(options)

**i Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates an attribute node of type ATTRIBUTE_NODE with the optional specified value and returns the new <a href="#">xml.Attr</a> object.  The localName, prefix, and namespaceURI properties of the new node are set to <b>null</b> .
---------------------------	--

<b>Returns</b>	xml.Attr
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description
options.name	string	Required	Name of the new attribute node.
options.value	string	Optional	Value for the attribute node. If unspecified, the value is an empty string.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.	Attribute with the specified name or value cannot be created.

## Syntax

```
//Add additional code
...
var attr = xmlDocument.createAttribute({
    name : 'lang',
    value : 'fr'
});
...
//Add additional code
```

## Document.createAttributeNS(options)

<b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.
--

<b>Method Description</b>	Creates an attribute node of type ATTRIBUTE_NODE, with the specified namespace value and optional specified value, and returns the new <a href="#">xml.Attr</a> object.  The <a href="#">Node.localName</a> , <a href="#">Node.prefix</a> , and <a href="#">Node.namespaceURI</a> properties of the new node are set to <b>null</b> .
<b>Returns</b>	xml.Attr

<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

<b>Note:</b> The options parameter is a JavaScript object.
--

Parameter	Type	Required / Optional	Description
options.namespaceURI	string	Required	Namespace URI of the attribute to create. Value can be <b>null</b> .
options.qualifiedName	string	Required	Qualified name of the new attribute node.
options.value	string	Optional	Value for the attribute node. If unspecified, the value is an empty string.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.	Attribute with the specified value cannot be created.

## Syntax

```
//Add additional code
...
var attr = xmlDocument.createAttributeNS({
    namespaceURI : "*",
    qualifiedName : 'lang',
    value : 'fr'
});
...
//Add additional code
```

## Document.createCDATASection(options)

<b>Note:</b> The content in this help topic pertains to SuiteScript 2.0.
--

<b>Method Description</b>	Creates a CDATA section node of type DOCUMENT_FRAGMENT_NODE with the specified data and returns the new <a href="#">xml.Node</a> object.
<b>Returns</b>	<a href="#">xml.Node</a>

<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.data	string	Required	Data for the new CDATA section node.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	Invalid argument type, expected string: data	Cannot create CDATA section node with the specified data.

## Syntax

```
//Add additional code
...
var newNode = xmlDocument.createCDATASection({
    data : 'Limited Edition.'
});
...
//Add additional code
```

## Document.createComment(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a Comment node of type COMMENT_NODE with the specified string.
<b>Returns</b>	<a href="#">xml.Node</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.data	string	Required	Data for the Comment node.

## Syntax

```
//Add additional code
...
var newNode = xmlDocument.createComment({
    data : 'This is a comment.'
});
...
//Add additional code
```

## Document.createDocumentFragment()

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a node of type DOCUMENT_FRAGMENT_NODE and returns the new <a href="#">xml.Node</a> object.
<b>Returns</b>	<a href="#">xml.Node</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var newNode = xmlDocument.createDocumentFragment();
...
//Add additional code
```

## Document.createElement(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a new node of type ELEMENT_NODE with the specified name and returns the new <a href="#">xml.Element</a> node.
---------------------------	---

	The <code>Node.localName</code> , <code>Node.prefix</code> , and <code>Node.namespaceURI</code> properties of the new node are set to <code>null</code> .
<b>Returns</b>	<code>xml.Element</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.tagName</code>	string	Required	Name of the element to create.

## Errors

Error Code	Message	Thrown If
<code>SSS_XML_DOM_EXCEPTION</code>	<code>INVALID_CHARACTER_ERR</code> : An invalid or illegal XML character is specified.	Element cannot be created with the specified tagName value.

## Syntax

```
//Add additional code
...
var elem = xmlDocument.createElement({
    tagName : 'book'
});
...
//Add additional code
```

## Document.createElementNS(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a new node of type ELEMENT_NODE with the specified namespace URI and name and returns the new <code>xml.Element</code> object.  The <code>Node.localName</code> , <code>Node.prefix</code> , and <code>Node.namespaceURI</code> properties of the new node are set to <code>null</code> .
<b>Returns</b>	<code>xml.Element</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Governance</b>	None
<b>Module</b>	N/xml Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.namespaceURI	string	Required	Namespace URI of the element to create. Can be null.
options.qualifiedName	string	Required	Qualified name of the element to create.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.	Element with the specified namespace cannot be created.

## Syntax

```
//Add additional code
...
var elem = xmlDocument.createElementNS({
    namespaceURI : '*',
    qualifiedName : 'book'
});
...
//Add additional code
```

## Document.createProcessingInstruction(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a new node of type PROCESSING_INSTRUCTION_NODE with the specified target and data and returns the new <a href="#">xml.Node</a> object.  The following example shows a sample processing instruction:  <?xml version="1.0"?>  Use a processing instruction node to keep processor-specific information in the text of the XML document.
<b>Returns</b>	<a href="#">xml.Node</a>
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Governance</b>	None
<b>Module</b>	N/xml Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.target	string	Required	Target part of the processing instruction.
options.data	string	Required	Data for the processing instruction.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.	Processing instruction node cannot be created with the specified target or data.

## Syntax

```
//Add additional code
...
var newNode = xmlDocument.createProcessingInstruction({
    target : 'xml'
    data : 'version="1.0"'
});
...
//Add additional code
```

## Document.createTextNode(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Creates a new text node and returns the new <a href="#">xml.Node</a> object.
<b>Returns</b>	<a href="#">xml.Node</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	N/xml Module
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.data	string	Required	Data for the text node.

## Syntax

```
//Add additional code
...
var newNode = xmlDocument.createTextNode({
    data : 'Sample Title'
});
...
//Add additional code
```

## Document.getElementById(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the element that has an ID attribute with the specified value as an <a href="#">xml.Element</a> object. Returns <b>null</b> if no such element exists.
<b>Returns</b>	<a href="#">xml.Element</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.elementId	string	Required	Unique ID value for an element.

## Syntax

```
//Add additional code
...
var elem = xmlDocument.getElementById({
    elementId : 'id12345'
});
```

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

## Document.getElementsByTagName(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns an array of <a href="#">xml.Element</a> objects with a specific tag name, in the order in which they appear in the XML document.
<b>Returns</b>	<a href="#">xml.Element[]</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
options.tagName	string	Required	Case-sensitive tag name of the element to match on. Use the * wildcard to match all elements.

### Syntax

```
//Add additional code
...
var elem = xmlDocument.getElementsByTagName({
    tagName : 'book'
});
...
//Add additional code
```

## Document.getElementsByTagNameNS(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns an array of <a href="#">xml.Element</a> objects with a specific tag name and namespace, in the order in which they appear in the XML document.
<b>Returns</b>	<a href="#">xml.Element[]</a>



**Important:** This method is not supported on Internet Explorer.

<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.namespaceURI	string	Required	Namespace URI to match on. Use the * wildcard to match all namespaces.
options.localName	string	Required	Localname property to match on. Use the * wildcard to match all local names.

## Syntax

```
//Add additional code
...
var elem = xmlDocument.getElementsByTagNameNS({
    namespaceURI : '*',
    localName : 'book'
});
...
//Add additional code
```

## Document.importNode(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Imports a node from another document to this document. This method creates a new copy of the source node.  If the <b>deep</b> parameter is set to <b>true</b> , it imports all children of the specified node. If set to <b>false</b> , it imports only the node itself.  Method returns the imported <a href="#">xml.Node</a> object.
<b>Returns</b>	<a href="#">xml.Node</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.importedNode	xml.Node	Required	Node from another XML document to import.
options.deep	boolean <code>true</code>   <code>false</code>	Required	Use <code>true</code> to import the node, its attributes, and all descendants. Use <code>false</code> to only import the node and its attributes.   <b>Important:</b> This parameter is not supported on Internet Explorer.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	NOT_SUPPORTED_ERR: The implementation does not support the requested type of object or operation.	Node cannot be imported.

## Syntax

```
...
var importedNode = xmlDocument1.importNode({
    importedNode : foreignNode,
    deep : true
});
...
```

## Document.doctype

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The doctype of the XML document.   <b>Important:</b> This property is not supported on Internet Explorer.
<b>Type</b>	xml.Element (read-only)
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/xml Module
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var doctype = xmlDocument.doctype;
...
//Add additional code
```

## Document.documentElement

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Root node of the XML document.  Use this property to directly access the <a href="#">xml.Element</a> object that represents the root node of an XML document.
<b>Type</b>	<a href="#">xml.Element</a> (read-only)
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var root = xmlDocument.documentElement;
...
//Add additional code
```

## Document.documentElement

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Location of the document or <code>null</code> if undefined.
<b>Type</b>	string
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
```

```
...
var documentURI = xmlDocument.documentElement;
...
//Add additional code
```

## Document.inputEncoding

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	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: <code>&lt;?xml version="1.0" encoding="UTF-8" standalone="yes"?&gt;</code>
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Syntax

```
//Add additional code
...
var encoding = xmlDocument.inputEncoding;
...
//Add additional code
```

## Document.xmlEncoding

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Part of the XML declaration, the XML encoding of the XML document. In the following declaration, the xmlEncoding property is UTF-8: <code>&lt;?xml version="1.0" encoding="UTF-8" standalone="yes"?&gt;</code>
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .

<b>Module</b>	N/xml Module
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var encoding = xmlDocument.xmlEncoding;
...
//Add additional code
```

## Document.xmlStandalone

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Part of the XML declaration, returns <b>true</b> if the current XML document is standalone or returns <b>false</b> if it is not.  In the following declaration, the <code>xmlStandalone</code> property is <b>true</b> : <code>&lt;?xml version="1.0" encoding="UTF-8" standalone="yes"?&gt;</code>
<b>Type</b>	boolean <b>true</b>   <b>false</b>
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	N/xml Module
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
...
var isStandalone = xmlDocument.xmlStandalone;
...
//Add additional code
```

## Document.xmlVersion

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Part of the XML declaration, the version number of the XML document.  In the following declaration, the <code>xmlVersion</code> property is 1.0: <code>&lt;?xml version="1.0" encoding="UTF-8" standalone="yes"?&gt;</code>
-----------------------------	--

	 <b>Important:</b> This property is not supported on Internet Explorer or Firefox.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	The implementation does not support the requested type of object or operation.	Cannot edit the XML version for the document.

## Syntax

```
//Add additional code
...
var version = xmlDocument.xmlVersion;
...
//Add additional code
```

## xml.Element



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	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:  <year>2015</year>  For a list of this object's methods and properties, see <a href="#">Element Object Members</a> .  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 <a href="#">Node Object Members</a> .
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Syntax

```
//Add additional code
```

```

...
var elem = parentNode[0].getElementsByTagName({tagName : 'title'});
...
//Add additional code

```

## Element.getAttribute(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns the value of the specified attribute.
<b>Returns</b>	<a href="#">xml.Attr</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
options.name	string	Required	Name of the attribute for which to return the value.

### Syntax

```

//Add additional code
...
var attr = elem[0].getAttribute({
    name : 'lang'
});
...
//Add additional code

```

## Element.getAttributeNode(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Retrieves an attribute node by name.
	<b>Important:</b> This method is not supported on Internet Explorer.
<b>Returns</b>	<a href="#">xml.Attr</a>
<b>Supported Script Types</b>	All script types

	For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.name	string	Required	The name of the attribute to return.

## Syntax

```
//Add additional code
...
var attr = elem[0].getATTRIBUTENode({
    name : 'lang'
});
...
//Add additional code
```

## Element.getAttributeNodeNS(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns an attribute node with the specified namespace URI and local name.  <b>Important:</b> This method is not supported on Internet Explorer.
<b>Returns</b>	string
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.namespaceURI	string	Required	Namespace URI of the attribute to return. Value can be <b>null</b> .

Parameter	Type	Required / Optional	Description
options.localName	string	Required	Local name of the attribute to return.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	Invalid argument type, expected string: namespaceURI	Attribute node with the specified namespace cannot be retrieved.

## Syntax

```
//Add additional code
...
var attr = elem[0].getAttributeNodeNS({
    namespaceURI : '*'
    localName : 'lang'
});
...
//Add additional code
```

## Element.getAttributeNS(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns an attribute value with the specified namespace URI and local name.
	 <b>Important:</b> This method is not supported on Internet Explorer.
<b>Returns</b>	string
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.namespaceURI	string	Required	Namespace URI of the attribute to return. Value can be <code>null</code> .
options.localName	string	Required	Local name of the attribute to return.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	Invalid argument type, expected string: namespaceURI	Attribute with the specified namespace cannot be retrieved.

## Syntax

```
//Add additional code
...
var attr = elem[0].getAttributeNS({
    namespaceURI : '*'
    localName : 'lang'
});
...
//Add additional code
```

## Element.getElementsByTagName(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns an array of descendant <a href="#">xml.Element</a> objects with a specific tag name, in the order in which they appear in the XML document.
<b>Returns</b>	<a href="#">xml.Element[]</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.tagName	string	Required	Case-sensitive tag name of the element to match on. Use the * wildcard to match all elements.

## Syntax

```
//Add additional code
...
var elem = parentNode[0].getElementsByTagName({tagName : 'title'});
...
```

```
//Add additional code
```

## Element.getElementsByTagNameNS(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns an array of descendant <code>xml.Element</code> objects with a specific tag name and namespace, in the order in which they appear in the XML document.
	 <b>Important:</b> This method is not supported on Internet Explorer.
<b>Returns</b>	<code>xml.Element[]</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.namespaceURI</code>	string	Required	Namespace URI to match on. Use the * wildcard to match all namespaces.
<code>options.localName</code>	string	Required	Localname property to match on. Use the * wildcard to match all local names.

### Errors

Error Code	Message	Thrown If
<code>SSS_XML_DOM_EXCEPTION</code>	Invalid argument type, expected string: namespaceURI	Elements with the specified namespace cannot be retrieved.

### Syntax

```
//Add additional code
...
var elem = parentNode[0].getElementsByTagNameNS({
    namespaceURI : '*',
    localName : 'lang'
});
...
//Add additional code
```

## Element.hasAttribute(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns <code>true</code> if the current element has an attribute with the specified name or if that attribute has a default value. Otherwise, returns <code>false</code> .
	<b>Important:</b> This method is not supported on Internet Explorer.
<b>Returns</b>	<code>boolean true   false</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description
<code>options.name</code>	string	Required	Name of the attribute to match on.

### Syntax

```
//Add additional code
...
var attrExists = elem[0].hasAttribute({
    name : 'lang'
});
...
//Add additional code
```

## Element.hasAttributeNS(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Returns <code>true</code> if the current element has an attribute with the specified local name and namespace or if that attribute has a default value. Otherwise, returns <code>false</code> .
	<b>Important:</b> This method is not supported on Internet Explorer.
<b>Returns</b>	<code>boolean true   false</code>

<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.namespaceURI	string	Required	Namespace URI of the attribute to match on.
options.localName	string	Required	Local name of the attribute to match on.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	Invalid argument type, expected string: namespaceURI	The method is called with an illegal namespace value.

## Syntax

```
//Add additional code
...
var attrExists = elem[0].hasAttributeNS({
    namespaceURI : '*',
    localName : 'lang'
});
...
//Add additional code
```

## Element.removeAttribute(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Removes the attribute with the specified name.
<b>Returns</b>	void
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>

<b>Since</b>	2015.2
--------------	--------

## Parameters

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

Parameter	Type	Required / Optional	Description
options.name	string	Required	Name of the attribute to remove.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	Invalid argument type, expected string: name	Attribute with the specified name cannot be removed.

## Syntax

```
//Add additional code
...
elem[0].removeAttribute({
    name : 'lang'
});
...
//Add additional code
```

## Element.removeAttributeNode(options)

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Removes the attribute specified as a <a href="#">xml.Attr</a> object.
<b>Returns</b>	<a href="#">xml.Attr</a>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.oldAttr	<a href="#">xml.Attr</a>	Required	<a href="#">xml.Attr</a> object to remove.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	NOT_FOUND_ERR: An attempt is made to reference a node in a context where it does not exist.	Attribute node cannot be removed.

## Syntax

```
//Add additional code
...
var removedAttr = elem[0].removeAttributeNode({
    oldAttr : attr
});
...
//Add additional code
```

## Element.removeAttributeNS(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Removes the attribute with the specified namespace URI and local name.
	 <b>Important:</b> This method is not supported on Internet Explorer.
<b>Returns</b>	void
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description
options.namespaceURI	string	Required	Namespace URI of the attribute node to remove.
options.localName	string	Required	Local name of the attribute node to remove.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	Invalid argument type, expected string: namespaceURI	Attribute with the specified namespace cannot be removed.

## Syntax

```
//Add additional code
...
elem[0].removeAttributeNS({
  namespaceURI : '*',
  localName : 'lang'
});
...
//Add additional code
```

## Element.setAttribute(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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.
<b>Returns</b>	void
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters



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

Parameter	Type	Required / Optional	Description
options.name	string	Required	Name of the attribute to add.
options.value	string	Required	Value of the attribute to add.

## Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.	Value for the attribute cannot be set.

## Syntax

```
//Add additional code
```

```

...
elem[0].setAttribute({
    name : 'lang',
    value : 'fr'
});
...
//Add additional code

```

## Element.setAttributeNode(options)



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <code>xml.Attr</code> object.
<b>Returns</b>	<code>xml.Attr</code>
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Parameters



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

Parameter	Type	Required / Optional	Description
<code>options.newAttr</code>	<code>xml.Attr</code>	Required	New <code>xml.Attr</code> object to add to the <code>xml.Element</code> object.

### Errors

Error Code	Message	Thrown If
<code>SSS_XML_DOM_EXCEPTION</code>	<code>INUSE_ATTRIBUTE_ERR</code> : An attempt is made to add an attribute that is already in use elsewhere.	Attribute node cannot be added.

### Syntax

```

//Add additional code
...
elem[0].setAttributeNode({
    newAttr : attr
});

```

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

## Element.setAttributeNodeNS(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <code>xml.Attr</code> object.
	 <b>Important:</b> This method is not supported on Internet Explorer.
<b>Returns</b>	<code>xml.Attr</code>
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.newAttr</code>	<code>xml.Attr</code>	Required	New <code>xml.Attr</code> object to add to the <code>xml.Element</code> object.

### Errors

Error Code	Message	Thrown If
<code>SSS_XML_DOM_EXCEPTION</code>	<code>INUSE_ATTRIBUTE_ERR</code> : An attempt is made to add an attribute that is already in use elsewhere.	Attribute node cannot be added.

### Syntax

```
//Add additional code
...
elem[0].setAttributeNode({
    newAttr : attr
});
...
//Add additional code
```

## Element.setAttributeNS(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	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 <a href="#">xml.Attr</a> object.
<b>Returns</b>	void
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Parameters

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

Parameter	Type	Required / Optional	Description
options.namespaceURI	string	Required	Namespace URI of the attribute node to add.
options.qualifiedName	string	Required	Fully qualified attribute name to add.
options.value	string	Required	String value of the attribute to add.

### Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.	Attribute node with the specified value cannot be added.

### Syntax

```
//Add additional code
...
elem[0].setAttributeNS({
    namespaceURI : '*',
    qualifiedName : 'lang',
    value : 'fr'
});
...
```

```
//Add additional code
```

## Element.tagName



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The tag name of this <a href="#">xml.Element</a> object.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Syntax

```
//Add additional code
...
var tagName = elem[0].tagName; \\ returns 'title'.
...
//Add additional code
```

## xml.Attr



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Object Description</b>	Represents an attribute node of an <a href="#">xml.Element</a> object. For a complete list of this object's properties, see <a href="#">Attr Object Members</a> .
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Syntax

```
//Add additional code
...
var attr = elem[0].getATTRIBUTENode({
    name : 'lang'
});
...
//Add additional code
```

## Attr.name



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	The name of an attribute.  This property is a qualified name if the <code>Node.localName</code> property for the parent <code>xml.Element</code> object is null.
<b>Type</b>	string (read-only)
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Syntax

```
//Add additional code
...
var attrName = attr.name; \\ returns 'lang'.
...
//Add additional code
```

## Attr.ownerElement



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<code>xml.Element</code> object that is the parent of the <code>xml.Attr</code> object. Value is <code>null</code> if the attribute is not used by an element.
	<b>Important:</b> This property is not supported on Internet Explorer.
<b>Type</b>	<code>xml.Element</code> (read-only)
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Syntax

```
//Add additional code
...
var attrElement = attr.ownerElement; \\ returns the title element.
...
//Add additional code
```

## Attr.specified

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	Returns <code>true</code> if the attribute value is set in the parsed XML document, and <code>false</code> if it is a default value in a DTD or Schema.
<b>Type</b>	boolean <code>true</code>   <code>false</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Syntax

```
//Add additional code
...
var attrSpecified = attr.specified;
...
//Add additional code
```

## Attr.value

**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Property Description</b>	<p>Value of an attribute. The value of the attribute is returned as a string.</p> <p>Character and general entity references are replaced with their values. For example, a character reference such as <code>&amp;#160;</code>; or an entity reference such as <code>&amp;nbsp;</code>; is replaced with a non-breaking space.</p> <p><b>Note:</b> 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.</p>
<b>Type</b>	<code>string</code>
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Errors

Error Code	Message	Thrown If
<code>SSS_XML_DOM_EXCEPTION</code>	Invalid argument type, expected string: value	Cannot set the attribute value with the specified value.

## Syntax

```
//Add additional code
...
var attrValue = attr.value;
...
//Add additional code
```

## xml.escape(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Prepares a string for use in XML by escaping XML markup, such as angle brackets, quotation marks, and ampersands.
<b>Returns</b>	string
<b>Supported Script Types</b>	All script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.xmlText	string	Required	String being escaped.	2015.2

## Syntax

```
//Add additional code
...
var xmlEscapedDocument = xml.escape({
    xmlText : xmlFileContent
});
...
//Add additional code
```

## xml.validate(options)

 **Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Method Description</b>	Validates an XML document against an XML Schema (XSD).
---------------------------	--

	 <b>Important:</b> This method only validates XML Schema (XSD); validation of other XML schema languages is not supported.
	The XML document must be passed as an <a href="#">xml.Document</a> 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.
<b>Returns</b>	void
<b>Supported Script Types</b>	All server-side script types For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Governance</b>	None
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

## Parameters

 <b>Note:</b>	The options parameter is a JavaScript object.
--	---

Parameter	Type	Required / Optional	Description	Since
options.xml	<a href="#">xml.Document</a>	Required	The <a href="#">xml.Document</a> object to validate.	2015.2
options.xsdFilePathOrId	number   string	Required	The file ID or path to the XSD in the File Cabinet to validate the XML document against.	2015.2
options.importFolderPathOrId	number   string	Optional	The folder ID or path to a folder in the File Cabinet containing additional XSD schemas which are imported by the parent XSD.	2015.2

## Errors

Error Code	Thrown If
SSS_XML_DOES_NOT_CONFORM_TO_SCHEMA	The provided XML is invalid for the provided schema.
SSS_INVALID_XML_SCHEMA_OR_DEPENDENCY	Schema is an incorrectly structured XSD or the dependent schema cannot be found.

## Syntax

```
//Add additional code
...
xml.validate({
    xml : xmlDocument,
    xsdFilePathOrId : 'SuiteScripts/schema_parent.xsd',
    importFolderPathOrId : 'SuiteScripts/'
});
...
//Add additional code
```

## xml.NodeType



**Note:** The content in this help topic pertains to SuiteScript 2.0.

<b>Enum Description</b>	Enumeration that holds the string values for the supported node types. The <a href="#">Node.nodeType</a> 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.
	<p><b>Note:</b> Enum values are constants and therefore read-only.</p>
<b>Supported Script Types</b>	All script types  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a> .
<b>Module</b>	<a href="#">N/xml Module</a>
<b>Since</b>	2015.2

### Values

<ul style="list-style-type: none"> <li>■ ATTRIBUTE_NODE</li> <li>■ CDATA_SECTION_NODE</li> <li>■ COMMENT_NODE</li> <li>■ DOCUMENT_FRAGMENT_NODE</li> </ul>	<ul style="list-style-type: none"> <li>■ DOCUMENT_NODE</li> <li>■ DOCUMENT_TYPE_NODE</li> <li>■ ELEMENT_NODE</li> <li>■ ENTITY_NODE</li> </ul>	<ul style="list-style-type: none"> <li>■ ENTITY_REFERENCE_NODE</li> <li>■ NOTATION_NODE</li> <li>■ PROCESSING_INSTRUCTION_NODE</li> <li>■ TEXT_NODE</li> </ul>
--	--	--

### Syntax

```
//Add additional code
...
var DocType = xmlDocument.nodeType; \\ returns DOCUMENT_NODE
...
//Add additional code
```