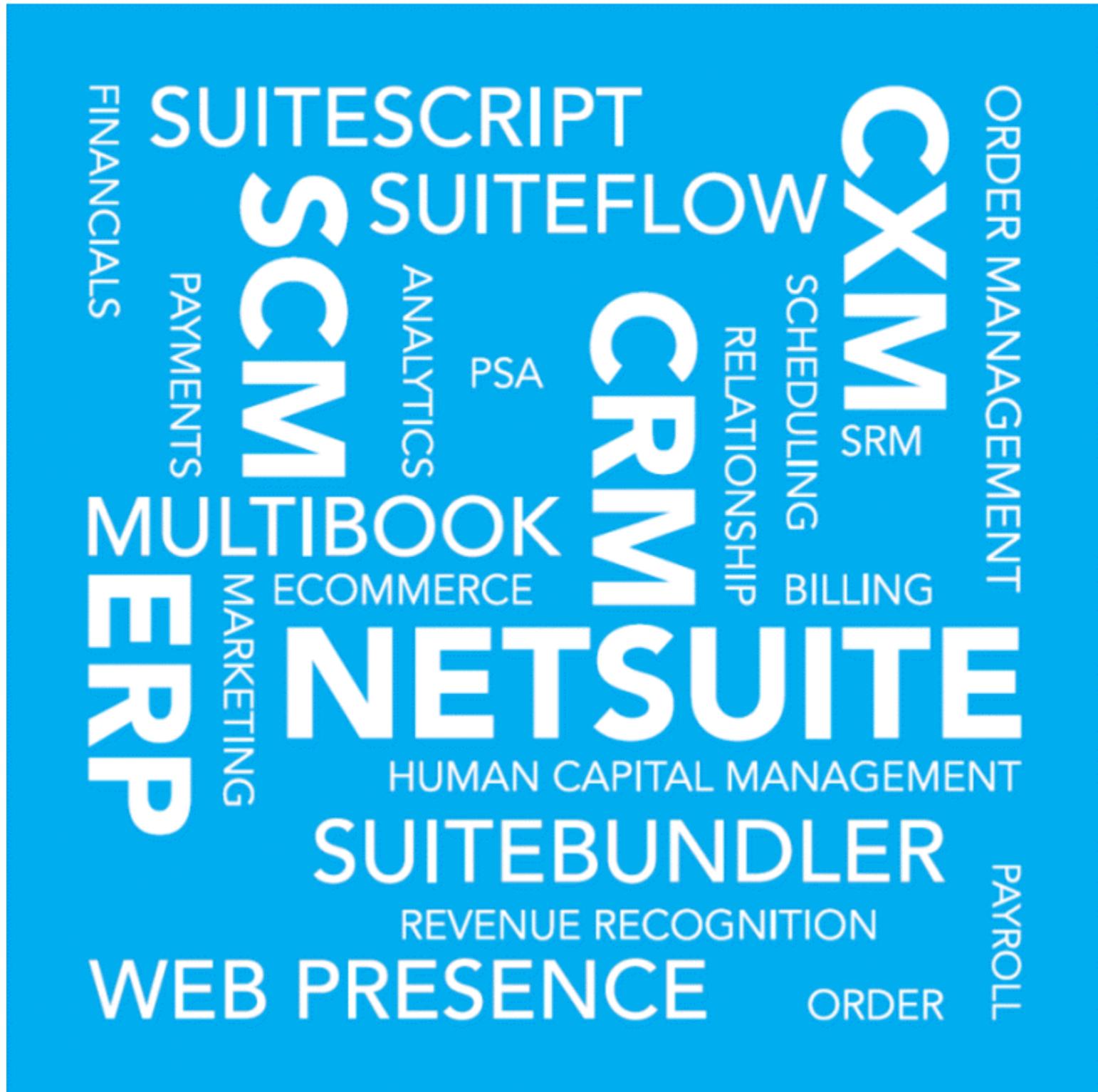


SuiteScript 2.0 API



General Notices

Sample Code

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

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

No Excessive Use of the Service

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

Beta Features

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

Integration with Third Party Applications

Oracle may make available to Customer certain features designed to interoperate with third party applications. To use such features, Customer may be required to obtain access to such third party applications from their providers, and may be required to grant Oracle access to Customer's account(s) on such third party applications. Oracle cannot guarantee the continued availability of such Service features or integration, and may cease providing them without entitling Customer to any refund, credit, or other compensation, if for example and without limitation, the provider of a third party application ceases to make such third party application generally available or available for interoperation with the corresponding Service features or integration in a manner acceptable to Oracle.

Copyright

This document is the property of Oracle, and may not be reproduced in whole or in part without prior written approval of Oracle. For Oracle trademark and service mark information for the Service, see www.netsuite.com/portal/company/trademark.shtml.

Table of Contents

SuiteScript 2.0 API Introduction	1
SuiteScript 2.0 – Getting Started	1
SuiteScript 2.0 – Script Architecture	2
SuiteScript 2.0 – Syntax	4
SuiteScript 2.0 – Script Creation Process	5
SuiteScript 1.0 to SuiteScript 2.0 API Map	6
SuiteScript 1.0 to SuiteScript 2.0 API Map – Functions (nlapi)	6
SuiteScript 1.0 to SuiteScript 2.0 API Map – Objects (nlobj)	16
SuiteScript 2.0 Script Types and Entry Points	43
Bundle Installation Script Type	45
Bundle Installation Script Entry Points	46
Client Script Type	48
Client Script Entry Points	55
Map/Reduce Script Type	62
Map/Reduce Stages	68
Map/Reduce Script Governance	69
Check the Status of a Map/Reduce Script	70
Map/Reduce Script Troubleshooting	71
Map/Reduce Script Type: Best Practices	72
Handling Server Restarts in Map/Reduce Scripts	73
Map/Reduce Script Entry Points	80
Map/Reduce Script API	85
Mass Update Script Type	111
Mass Update Script Entry Points	111
Portlet Script Type	112
render(params)	117
Portlet Object	117
RESTlet Script Type	127
Getting Started with RESTlets	127
RESTlet Authentication	133
RESTlet Reference	141
RESTlet Script and Request Samples	147
RESTlet Script Entry Points	154
Scheduled Script Type	157
Scheduled Script Execution	159
Scheduled Script Deployment	159
Scheduled Script Status and Monitoring	162
execute	163
context.InvocationType	163
Best Practice: Handling Server Restarts in Scheduled Scripts (SuiteScript 2.0)	164
Suitelet Script Type	167
onRequest(params)	170
User Event Script Type	170
User Event Script API	174
Workflow Action Script Type	177
onAction(scriptContext)	179
SuiteScript 2.0 Global Objects and Methods	180
Module Dependency Paths	180
define Function	181
define([dependencies,] callback)	183
define(id, [dependencies,] callback)	185
define(callback)	187
define(moduleobject)	188

require Function	189
require([dependencies], callback)	190
require Configuration	192
log Object	193
util Object	193
toString()	194
JSON object	195
JSON.parse(text)	195
JSON.stringify(obj)	196
Promise object	196
SuiteScript 2.0 Modules	200
N/auth Module	202
auth.changeEmail(options)	203
auth.changePassword(options)	204
N/cache Module	205
cache.Cache	208
cache.getCache(options)	212
cache.Scope	213
N/config Module	214
config.load(options)	215
config.Type	217
N/crypto Module	218
crypto.Cipher	222
crypto.CipherPayload	225
crypto.Decipher	226
crypto.Hash	228
crypto.Hmac	230
crypto.SecretKey	232
crypto.createCipher(options)	234
crypto.createDecipher(options)	235
crypto.createHash(options)	236
crypto.createHmac(options)	236
crypto.createSecretKey(options)	237
crypto.EncryptionAlg	238
crypto.HashAlg	239
crypto.Padding	240
N/currency Module	240
currency.exchangeRate(options)	241
N/currentRecord Module	243
currentRecord.CurrentRecord	250
currentRecord.Field	296
currentRecord.get()	304
currentRecord.get.promise()	305
N/email Module	306
email.send(options)	307
email.send.promise(options)	312
email.sendBulk(options)	312
email.sendBulk.promise(options)	316
email.sendCampaignEvent(options)	316
email.sendCampaignEvent.promise(options)	318
N/encode Module	318
encode.convert(options)	319
encode.Encoding	320
N/error Module	321
error.SuiteScriptError	323

error.UserEventError	326
error.create(options)	330
N/file Module	331
file.File	335
file.create(options)	349
file.delete(options)	352
file.load(options)	353
file.Encoding	354
file.Type	355
N/format Module	356
format.format(options)	357
format.parse(options)	359
format.Type	361
format.Timezone	361
N/http Module	365
http.ClientResponse	369
http.ServerRequest	372
http.ServerResponse	378
http.get(options)	389
http.get.promise(options)	390
http.delete(options)	391
http.delete.promise(options)	392
http.request(options)	393
http.request.promise(options)	394
http.post(options)	395
http.post.promise(options)	396
http.put(options)	397
http.put.promise(options)	398
http.CacheDuration	399
http.Method	400
http.RedirectType	401
N/https Module	401
https.SecureString	407
https.createSecureKey(options)	412
https.createSecureKey.promise(options)	412
https.createSecureString(options)	413
https.createSecureString.promise(options)	414
https.ClientResponse	415
https.ServerRequest	417
https.ServerResponse	423
https.get(options)	434
https.get.promise(options)	435
https.delete(options)	436
https.delete.promise(options)	437
https.request(options)	438
https.request.promise(options)	439
https.post(options)	440
https.post.promise(options)	441
https.put(options)	442
https.put.promise(options)	443
https.CacheDuration	444
https.Encoding	445
https.Method	445
N/log Module	446
log.audit(options)	449

log.debug(options)	450
log.emergency(options)	451
log.error(options)	452
N/plugin Module	453
plugin.findImplementations(options)	454
plugin.loadImplementation(options)	455
N/portlet Module	456
portlet.resize	458
portlet.refresh	459
N/record Module	459
record.Column	478
record.Field	480
record.Record	484
record.Sublist	543
record.attach(options)	546
record.attach.promise(options)	548
record.copy(options)	549
record.copy.promise(options)	551
record.create(options)	553
record.create.promise(options)	555
record.delete(options)	557
record.delete.promise(options)	558
record.detach(options)	559
record.detach.promise(options)	560
record.load(options)	562
record.load.promise(options)	564
record.submitFields(options)	565
record.submitFields.promise(options)	567
record.transform(options)	569
record.transform.promise(options)	573
record.Type	575
N/redirect Module	577
redirect.redirect(options)	578
redirect.toRecord(options)	579
redirect.toSavedSearch(options)	580
redirect.toSavedSearchResult(options)	581
redirect.toSearch(options)	582
redirect.toSearchResult(options)	583
redirect.toSuitelet(options)	583
redirect.toTaskLink(options)	584
N/render Module	585
render.EmailMergeResult	589
render.TemplateRenderer	591
render.bom(options)	599
render.create()	600
render.mergeEmail(options)	601
render.packingSlip(options)	602
render.pickingTicket(options)	603
render.statement(options)	604
render.transaction(options)	605
render.xmlToPdf(options)	606
render.DataSource	607
render.PrintMode	607
N/runtime Module	608
runtime.Script	613

runtime.Session	617
runtime.User	619
runtime.getCurrentScript()	626
runtime.getCurrentSession()	626
runtime.getCurrentUser()	627
runtime.isFeatureInEffect(options)	628
runtime.accountId	628
runtime.envType	629
runtime.executionContext	629
runtime.queueCount	630
runtime.version	630
runtime.ContextType	631
runtime.EnvType	631
runtime.Permission	632
N/search Module	632
search.Search	643
search.Result	653
search.Column	660
search.Filter	667
search.ResultSet	670
search.Page	675
search.PagedData	681
search.PageRange	685
search.create(options)	686
search.create.promise(options)	690
search.load(options)	690
search.load.promise(options)	692
search.delete(options)	692
search.delete.promise(options)	693
search.duplicates(options)	694
search.duplicates.promise(options)	696
search.global(options)	697
search.global.promise(options)	698
search.lookupFields(options)	698
search.lookupFields.promise(options)	700
search.createColumn(options)	701
search.createFilter(options)	702
search.Operator	704
search.Sort	705
search.Summary	706
search.Type	706
N/sftp Module	709
Setting up an SFTP Transfer	711
SFTP Authentication	712
Supported Cipher Suites and Host Key Types	714
Supported SuiteScript File Types	715
sftp.Connection	717
sftp.createConnection(options)	720
N/sso Module	722
sso.generateSuiteSignOnToken(options)	723
N/task Module	725
task.ScheduledScriptTask	735
task.ScheduledScriptTaskStatus	738
task.MapReduceScriptTask	741
task.MapReduceScriptTaskStatus	744

task.CsvImportTask	753
task.CsvImportTaskStatus	757
task.EntityDeduplicationTask	759
task.EntityDeduplicationTaskStatus	763
task.SearchTask	764
task.SearchTaskStatus	768
task.WorkflowTriggerTask	772
task.WorkflowTriggerTaskStatus	775
task.create(options)	776
task.checkStatus(options)	782
task.TaskType	783
task.TaskStatus	784
task.MasterSelectionMode	785
task.DedupeMode	785
task.DedupeEntityType	786
task.MapReduceStage	787
N/transaction Module	787
transaction.void(options)	789
transaction.void.promise(options)	790
transaction.Type	791
N/ui/dialog Module	792
dialog.alert(options)	794
dialog.confirm(options)	795
dialog.create(options)	796
N/ui/message module	798
message.Message	799
message.create(options)	801
message.Type	801
N/ui/serverWidget Module	802
serverWidget.Assistant	814
serverWidget.AssistantStep	839
serverWidget.Button	847
serverWidget.Field	850
serverWidget.FieldGroup	865
serverWidget.Form	869
serverWidget.List	896
serverWidget.ListColumn	905
serverWidget.Sublist	908
serverWidget.Tab	920
serverWidget.createAssistant(options)	922
serverWidget.createForm(options)	923
serverWidget.createList(options)	924
serverWidget.AssistantSubmitAction	925
serverWidget.FieldBreakType	925
serverWidget.FieldDisplayType	926
serverWidget.FieldLayoutType	927
serverWidget.FieldType	928
serverWidget.FormPageLinkType	929
serverWidget.LayoutJustification	930
serverWidget.ListStyle	931
serverWidget.SublistDisplayType	932
serverWidget.SublistType	933
N/url Module	934
url.format(options)	935
url.resolveDomain(options)	936

url.resolveRecord(options)	937
url.resolveScript(options)	938
url.resolveTaskLink(options)	939
url.HostType	940
N/util Module	941
util.isArray(obj)	943
util.isBoolean(obj)	944
util.isDate(obj)	945
utilisFunction(obj)	945
util.isNumber(obj)	946
utilisObject(obj)	947
util.isRegExp(obj)	947
util.isString(obj)	948
util.nanoTime()	949
util.each(iterable, callback)	949
util.extend(receiver, contributor)	950
N/workflow Module	952
workflow.initiate(options)	953
workflow.trigger(options)	954
N/xml Module	956
xml.Parser	965
xml.XPath	967
xml.Node	968
xml.Document	985
xml.Element	1000
xml.Attr	1014
xml.escape(options)	1016
xml.validate(options)	1017
xml.NodeType	1018
SuiteScript 2.0 JSDoc Validation	1020
Controlling Access to Scripts and Custom Modules	1023
SuiteScript 2.0 Entry Point Script Deployment	1026
Record-Level and Form-Level Scripts	1026
SuiteScript 2.0 Entry Point Script Validation	1027
Entry Point Script Validation Guidelines	1027
Entry Point Script Validation Examples	1030
Entry Point Script Validation Error Reference	1032
SuiteScript 2.0 Record-Level Scripts	1035
Script Record Creation	1035
Script Deployment	1039
Viewing System Notes	1042
SuiteScript 2.0 Form-Level Scripts	1042
Attaching a Client Script to a Form	1043
Configuring a Custom Action	1044
SuiteScript 2.0 Custom Modules	1046
Naming a Custom Module	1047
Custom Module Examples	1047
Troubleshooting Errors	1051
Frequently Asked Questions: Custom Modules	1051
SuiteScript 2.0 Scripting Records and Subrecords	1053
Record and Subrecord Scripting Overview	1053
Scripting Records	1053
Scripting Subrecords	1053
Understanding Subrecords	1053
Subrecord Scripting in SuiteScript 2.0 Compared With 1.0	1063

Scripting Subrecords that Occur on Sublist Lines	1064
Scripting Subrecords that Occur in Body Fields	1089

SuiteScript 2.0 API Introduction

- SuiteScript 2.0 – Getting Started
- SuiteScript 2.0 – Script Architecture
- SuiteScript 2.0 – Syntax
- SuiteScript 2.0 – Script Creation Process

SuiteScript 2.0 – Getting Started

Features Not Supported in SuiteScript 2.0

The following are not currently supported in SuiteScript 2.0:

- Core Plug-ins
- SSP scripts
- SuiteCommerce Advanced

SuiteScript Versioning

This release (SuiteScript 2.0) and all future releases of SuiteScript will maintain the following versioning system.

Version Type	Numbering Pattern	Description
Major	Version 2.0, 3.0, 4.0	Major versions of SuiteScript include significant functionality changes and improvements. Major versions are not backward compatible with previously released versions.
Minor	Version 3.1, 3.2, 3.3	Minor versions of SuiteScript include enhancements to existing features. Minor versions are backward compatible with all versions released since the last major version. For example, SuiteScript Version 3.2 is backward compatible with Versions 3.0 and 3.1. It is not backward compatible with Versions 1.0 or 2.0.
Patch	Does not apply	Patch versions of SuiteScript are included with regular NetSuite bug fix releases. Patch versions are backward compatible with all versions released since the last major version.

Version Cohabitation Rules

Your script (entry point script and supporting library scripts) must use either SuiteScript 1.0 or SuiteScript 2.0. You cannot use APIs from both versions in one script.

You can, however, have multiple scripts that use different SuiteScript versions. These can be deployed in the same account, in the same SuiteApp, and on the same record.

The map/reduce script type is a new functionality introduced with SuiteScript 2.0. You cannot use `nlapiScheduleScript(scriptId, deployId, params)` to schedule a Map/Reduce script. See [Map/Reduce Script Type](#) for more information.

SuiteScript 2.0 – Script Architecture

Module Loader

SuiteScript 2.0 is designed to be modular. SuiteScript 2.0 implements its modular architecture with the Asynchronous Module Definition (AMD) specification. AMD is used to define and load JavaScript modules and their dependencies. For additional information regarding AMD, see <http://requirejs.org/docs/whyamd.html>.

All SuiteScript 2.0 APIs are organized into modules. Each SuiteScript 2.0 module encapsulates a specific set of functionality. For example, you load the file module when you need to work with files in NetSuite. For additional information on SuiteScript 2.0 modules, see [SuiteScript 2.0 Modules](#).

define() Function

Use the `define()` function to load SuiteScript 2.0 modules and create custom modules. When you use the `define()` function, it loads all dependencies before it executes any logic.

In SuiteScript 2.0, the `define()` function returns an object that encapsulates the module you are defining.

 **Note:** If you need to ad hoc debug your code in the NetSuite Debugger, you must use a `require()` function. The NetSuite Debugger cannot step though a `define()` function.

You must use the `define()` function in your entry point script. The return statement must include at least one entry point and entry point function. All entry points must belong to the same script type. For additional information on script types and entry points, see [SuiteScript 2.0 Script Types and Entry Points](#).

In the user event script below, the return statement includes all three user event entry points.

```
/** 
 * @NApiVersion 2.x
 * @NScriptType UserEventScript
 */
define(['N/record'],
  function (record)
{
  function doSomething1(context)
  {
    if (context.type !== context.UserEventType.CREATE)
      return;
    var customerRecord = context.newRecord;
    customerRecord.setValue('phone', '555-555-5555');
    if (!customerRecord.getValue('salesrep'))
      customerRecord.setValue('salesrep', 46);
  }
  function doSomething2(context)
```

```

{
    if (context.type !== context.UserEventTypes.CREATE)
        return;
    var customerRecord = context.newRecord;
    customerRecord.setValue('comments', 'Please follow up with this customer!');
}
function doSomething3(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('title', '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('Call record created successfully', 'Id: ' + callId);
        }
        catch (e)
        {
            log.error(e.name);
        }
    }
    return {
        beforeLoad: doSomething1,
        beforeSubmit: doSomething2,
        afterSubmit: doSomething3
    };
});

```

require() Function

The `require()` function is used to load modules. When you use the `require()` function, dependencies are not loaded until they are needed. In the example below, the `record` module is loaded when `record.create()` is called.

The `require()` function does not return a value.

```
/**
 * @NApiVersion 2.x
 */
require(['N/record'],
    function(record)
{
    function createAndSaveContactRecord()
    {
        var nameData = {

```

```

        firstname: 'John',
        middlename: 'Doe',
        lastname: 'Smith'
    };
    var recordObj = record.create({
        type: record.Type.CONTACT,
        isDynamic: true
    });
    recordObj.setValue({
        fieldId: 'subsidiary',
        value: '1'
    });
    for (var key in nameData) {
        if(nameData.hasOwnProperty(key)) {
            recordObj.setValue({
                fieldId: key,
                value: nameData[key]
            });
        }
    }
    var recordId = recordObj.save({
        enableSourcing: false,
        ignoreMandatoryFields: false
    });
}
}

createAndSaveContactRecord();
});

```

SuiteScript 2.0 – Syntax

- All SuiteScript 2.0 methods take a plain JavaScript object as an input. In the example below, the method `record.load` takes in a JavaScript object that consists of two key/value pairs: `type: record.Type.SALES_ORDER` and `id: 6`.

```

var recObj = record.load({
    type: record.Type.SALES_ORDER,
    id: 6
});

```

Within the SuiteScript 2.0 API, all method inputs are named `options`. For example, the `record.load` signature is listed as `record.load(options)`.

Note: JavaScript objects can contain a mixture of value types. See the applicable SuiteScript 2.0 API help topic for supported value types.

- All SuiteScript 2.0 booleans take a value of `true` or `false`. All other boolean values (for example: `T` or `F`) throw an error.
- Parameter types in SuiteScript 2.0 are strictly adhered to. You must pass in valid parameter types, as listed in the SuiteScript 2.0 help. SuiteScript 2.0 does not convert invalid parameter values to valid values.
- Enumerations encapsulate common constants (for example, standard record types).

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

- Sublist and column indexing begins at 0.

SuiteScript 2.0 – Script Creation Process

The following is a very basic process flow for SuiteScript 2.0 script creation.

Note: Your specific process may vary, depending on the content of your script.

Stage	Description	Additional Information
1	Use the <code>define()</code> function to load SuiteScript 2.0 modules in your entry point script. Your entry point script is the script you attach to the script record.	SuiteScript 2.0 – Script Architecture SuiteScript 2.0 Global Objects and Methods
2	Add required JSDoc tags to your entry point script.	SuiteScript 2.0 JSDoc Validation
3	Add at least one entry point function to your entry point script. An entry point function is a named function that is executed when an entry point is triggered. Important: Your entry point script can implement only one script type. For example, your entry point script cannot return both a <code>beforeLoad</code> entry point and an <code>onRequest</code> entry point.	SuiteScript 2.0 Script Types and Entry Points
4	Organize your supporting code into custom modules (as a replacement for SuiteScript 1.0 libraries). Create these modules with the <code>define()</code> function and then load them in your entry point script.	SuiteScript 2.0 Global Objects and Methods
5	Upload and deploy your script to NetSuite.	SuiteScript 2.0 Entry Point Script Deployment

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.

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: 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
nlapiAddDays(d, days)	See Notes	See Notes	This API does not have a SuiteScript 2.0 equivalent. SuiteScript 2.0 is compatible with third party JavaScript APIs that provide this functionality (for example, Moment.js). For information on using third party APIs with SuiteScript 2.0, see SuiteScript 2.0 Custom Modules .
nlapiAddMonths(d, months)	See Notes	See Notes	This API does not have a SuiteScript 2.0 equivalent. SuiteScript 2.0 is compatible with third party JavaScript APIs that provide this functionality (for example, Moment.js). For information on importing third party APIs, see

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
			SuiteScript 2.0 Custom Modules.
nlapiAttachRecord(type, id, type2, id2, attributes)	record.attach(options)	N/record Module	<pre>var recordId = record.attach({ record: { type: record.Type.FILE, id: '447' }, to: { type: record.type.CUSTOMER, id: 530 } });</pre>
nlapiCancelLineItem(type)	Record.cancelLine(options) CurrentRecord.cancelLine(options)	N/record Module N/currentRecord Module	
nlapiCommitLineItem(type)	Record.commitLine(options) CurrentRecord.commitLine(options)	N/record Module N/currentRecord Module	For N/record script samples, see: <ul style="list-style-type: none"> ■ N/record Module Script Samples ■ Example: Creating an Inventory Detail Sublist Subrecord
nlapiCopyRecord(type, id, initializeValues)	record.copy(options)	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>
nlapiCreateAssistant(title, hideHeader)	serverWidget.createAssistant(options)	N/ui/serverWidget Module	
nlapiCreateCSVImport()	task.create(options)	N/task Module	For script samples, see N/task Module Script Sample .
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 Scripting Subrecords .
nlapiCreateEmailMerger(templateId)	render.mergeEmail(options)	N/render Module	
nlapiCreateError(code, details, suppressNotification)	error.create(options)	N/error Module	For a script sample, see N/error Module Script Sample .
nlapiCreateFile(name, type, contents)	file.create(options)	N/file Module	For a script sample, see N/file Module Script Sample .
nlapiCreateForm(title, hideNavbar)	serverWidget.createForm(options)	N/ui/serverWidget Module	For a script sample, see N/ui/serverWidget Module Script Sample
nlapiCreateList(title, hideNavbar)	serverWidget.createList(options)	N/ui/serverWidget Module	
nlapiCreateRecord(type, initializeValues)	record.create(options)	N/record Module	

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlapiCreateSearch(type, filters, columns)	search.create(options)	N/search Module	For a script sample, see N/search Module Script Samples .
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 Scripting Subrecords .
nlapiCreateTemplateRenderer()	render.create()	N/render Module	For a script sample, see N/render Module Script Sample .
nlapiDateToString(d, format)	format.format(options)	N/format Module	For a script sample, see N/format Module Script Sample
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 <code>isDisabled</code> is a property.
nlapiDisableLineItemField(type, fldnam, val)	Field.isDisabled	N/currentRecord Module	Note that <code>isDisabled</code> 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 Scripting Subrecords .
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 Scripting Subrecords .
nlapiEncrypt(s, algorithm, key)	See Notes	See Notes	For SuiteScript 2.0 encryption, hashing, and HMAC functionality, see the N/crypto Module module. For SuiteScript 2.0 encoding functionality, see the N/encode Module module.
nlapiEscapeXML(text)	xml.escape(options)	N/xml Module	
nlapiExchangeRate(sourceCurrency, targetCurrency, effectiveDate)	currency.exchangeRate(options)	N/currency Module	For a script sample, see N/currency Module Script Sample .
nlapiFindLineItemMatrixValue(type, fldnam, val, column)	Record.findMatrixSublistLineWithValue(options) CurrentRecord.findMatrixSublistLineWithValue(options)	N/record Module N/currentRecord Module	

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlapiFindLineItemValue(type, fldnam, val)	Record.findSublistLineWithVal(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 N/format module and not the N/currency module. For a script sample, see N/format Module Script Sample
nlapiGetContext()	runtime.getCurrentScript() runtime.getCurrentSession() runtime.getCurrentUser()	N/runtime Module	For a script sample, see N/runtime Module Script Sample .
nlapiGetCurrentLineItemDateValue(type, fieldId, timeZone)	See Notes	N/format Module	Use the N/format 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 N/format module to mimic this functionality in SuiteScript 2.0.
nlapiGetDepartment()	User.department	N/runtime Module	
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	

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
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 N/task Module Script Sample .
nlapiGetLineItemCount(type)	Record.getLineCount(options) CurrentRecord.getLineCount(options)	N/record Module N/currentRecord Module	
nlapiGetLineItemDateTimeValue(type, fieldId, lineNum, timeZone)	See Notes	N/format Module	Use the N/format 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	
nlapiGetLineItemMatrixField(type, fldnam, linenum, column)	Record.getMatrixSublistField(options) CurrentRecord.getMatrixSublistField(options)	N/record Module N/currentRecord Module	
nlapiGetLineItemMatrixValue(type, fldnam, linenum, column)	Record.getMatrixSublistValue(options) CurrentRecord.getMatrixSublistValue(options)	N/record Module N/currentRecord Module	
nlapiGetLineItemText(type, fldnam, linenum)	Record.getSublistText(options) CurrentRecord.getSublistText(options)	N/record Module N/currentRecord Module	
nlapiGetLineItemValue(type, fldnam, linenum)	Record.getSublistValue(options) CurrentRecord.getSublistValue(options)	N/record Module N/currentRecord Module	
nlapiGetLineItemValues(type, fldname, linenum)	Record.getSublistValue(options) CurrentRecord.getSublistValue(options)	N/record Module N/currentRecord Module	Method returns an array for multi-select fields.
nlapiGetLocation()	User.location	N/runtime Module	Note that <code>location</code> is a property.
nlapiGetLogin()	auth.changeEmail(options) auth.changePassword(options)	N/auth Module	For a script sample, see N/auth Module Script Sample .
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	

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
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 beforeLoad(scriptContext), beforeSubmit(scriptContext), or afterSubmit(scriptContext) user event script. <pre>function afterSubmit(context) { var newRec = context.newRecord; }</pre> <p>For additional information and a full script sample, see User Event Script Type</p>
nlapiGetOldRecord()	See Notes	See Notes	To mimic this functionality in SuiteScript 2.0, use the following code in a beforeSubmit(scriptContext) or afterSubmit(scriptContext) user event script. <pre>function afterSubmit(context) { var oldRec = context.oldRecord; }</pre> <p>For additional information and a full script sample, see User Event Script Type</p>
nlapiGetRecordId()	Record.id CurrentRecord.id	N/record Module N/currentRecord Module	
nlapiGetRecordType()	Record.type CurrentRecord.type	N/record Module N/currentRecord Module	
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 N/workflow Module Script Sample .
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	
nlapisLineItemChanged(type)	Sublist.isChanged	N/record Module	Note that isChanged is a property

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
			<code>record.getSublist("addressbook").isChanged</code>
nlapiLoadConfiguration(type)	config.load(options)	N/config Module	For a script sample, see N/config Module Script Sample .
nlapiLoadFile(id)	file.load(options)	N/file Module	For a script sample, see N/file Module Script Sample .
nlapiLoadRecord(type, id, initializeValues)	record.load(options)	N/record Module	
nlapiLoadSearch(type, id)	search.load(options)	N/search Module	For a script sample, see N/search Module Script Samples .
nlapiLogExecution(type, title, details)	log.audit(options) log.debug(options) log.emergency(options) log.error(options)	N/log Module	For a script sample, see log Module Script Sample .
nlapiLookupField(type, id, fields, text)	search.lookupFields(options)	N/search Module	
nlapiOutboundSSO(id)	sso.generateSuiteSignOnToken(options)		For a script sample, see N/sso Module Script Sample .
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 N/render Module Script Sample .
nlapiRefreshLineItems(type)	N/A	N/A	This API does not have a SuiteScript 2.0 equivalent.
nlapiRefreshPortlet()	portlet.refresh	N/portlet Module	For a script sample, see N/portlet Module Script Sample
nlapiRemoveCurrentLineItemSubrecord(sublist, fldname)	Record.removeCurrentLineItemSubrecord(options) CurrentRecord.removeCurrentLineItemSubrecord(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 Scripting Subrecords .
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 Scripting Subrecords .
nlapiRequestURL(url, postdata, headers, callback, httpMethod)	http.delete(options) http.get(options) http.post(options)	N/http Module	

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
	http.put(options) http.request(options)		
nlapiRequestURLWithC redentials(credentials, url, postdata, headers, httpsMethod)		https	
nlapiResizePortlet()	portlet.resize	N/portlet Module	For a script sample, see N/portlet Module Script Sample .
nlapiResolveURL(type, identifier, id, displayMode)	url.resolveRecord(options) url.resolveScript(options) url.resolveTaskLink(options)	N/url Module	For a script sample, see N/url Module Script Sample .
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 N/search Module Script Samples .
nlapiSelectLineItem(type, linenum)	Record.selectLine(options) CurrentRecord. selectLine(options)	N/record Module N/currentRecord Module	
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 XPath.select(options) and then inspect the Node.textContent property.
nlapiSelectValues(node, path)	See Notes	N/xml Module	To mimic this functionality in SuiteScript 2.0, select an array of nodes with XPath.select(options) and then loop through each node's Node.textContent property.
nlapiSendCampaignE mail(campaigneventid, recipientid)	email.sendCampaignEve nt(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 N/email Module Script Sample .

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlapiSendFax(author, recipient, subject, body, records, attachments)	N/A	N/A	This API does not have a SuiteScript 2.0 equivalent.
nlapiSetCurrentLineItemDateValue(type, fieldId, date, timeZone)	See Notes	N/format Module	Use the <code>N/format</code> module to mimic this functionality in SuiteScript 2.0.
nlapiSetCurrentLineItemMatrixValue(type, fldnam, column, value, firefieldchanged, synchronous)	Record.setCurrentMatrixSublistValue(options) CurrentRecord.setMatrixSublistValue(options)	N/record Module N/currentRecord Module	
nlapiSetCurrentLineItemText(type, fldnam, text, firefieldchanged, synchronous)	Record.setCurrentSublistText(options) CurrentRecord.setCurrentSublistText(options)	N/record Module N/currentRecord Module	
nlapiSetCurrentLineItemValue(type, fldnam, value, firefieldchanged, synchronous)	Record.setCurrentSublistValue(options) CurrentRecord.setCurrentSublistValue(options)	N/record Module N/currentRecord Module	
nlapiSetCurrentLineItemValues(type, fldnam, values, firefieldchanged, synchronous)	Record.setCurrentSublistValue(options) CurrentRecord.setCurrentSublistValue(options)	N/record Module N/currentRecord Module	
nlapiSetDateTimeValue(fieldId, date, timeZone)	See Notes	N/format Module	Use the <code>N/format</code> 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	
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, date, timeZone)	See Notes	N/format Module	Use the <code>N/format</code> module to mimic this functionality in SuiteScript 2.0.
nlapiSetLineItemValue(type, fldnam, lineNum, value)	Record.setSublistValue(options)	N/record Module	
nlapiSetMatrixValue(type, fldnam, column, value, firefieldchanged, synchronous)	Record.setMatrixHeaderValue(options) CurrentRecord.setMatrixHeaderValue(options)	N/record Module N/currentRecord Module	
nlapiSetRecoveryPoint()	See Notes	See Notes	The <code>Map/Reduce Script Type</code> automatically incorporates yielding.
nlapiSetRedirectURL(type, identifier, id, editmode, parameters)	redirect.redirect(options) redirect.toRecord(options)	N/redirect Module	For a script sample, see N/redirect Module Script Sample .

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
	redirect.toSuitelet(options) redirect.toTaskLink(options)		
nlapiStringToDate(str, format)	format.parse(options)	N/format Module	For a script sample, see N/format Module Script Sample .
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 N/file Module Script Sample .
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.getCurrentSublistSubrecord(options) Record.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 Scripting Subrecords .
nlapiViewLineitemSubrecord(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 Scripting Subrecords .
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 Scripting Subrecords .
nlapiVoidTransaction(transactionType, recordId)	transaction.void(options)	N/transaction Module	For a script sample, see N/transaction Module Script Sample .
nlapiXMLToPDF(xmlstring)	render.xmlToPdf(options)	N/render Module	Note that TemplateRenderer.renderAsPdf(

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
	TemplateRenderer.renderAsPdf()) is equivalent to nlapiXMLToPDF(nlobjEmailMerger.renderToString()). For a script sample, see N/render Module Script Sample .
nlapiXMLToString(xml)	Parser.toString(options)	N/xml Module	
nlapiYieldScript()	See Notes	See Notes	Note that the Map/Reduce Script Type 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.

 **Note:** To view a mapping of SuiteScript 1.0 Functions (prefixed with “nlapi”) to their corresponding SuiteScript 2.0 APIs, see [SuiteScript 1.0 to SuiteScript 2.0 API Map – Functions \(nlapi\)](#).

- nlobjAssistant
- nlobjButton
- nlobjButton
- nlobjColumn
- nlobjConfiguration
- nlobjContext
- nlobjCredentialBuilder
- nlobjCSVImport
- nlobjDuplicateJobRequest
- nlobjEmailMerger
- nlobjError
- nlobjField
- nlobjFieldGroup
- nlobjFile
- nlobjForm
- nlobjFuture
- nlobjJobManager
- nlobjList
- nlobjLogin
- nlobjMergeResult
- nlobjPortet

- 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
nlobjAssistant	serverWidget.Assistant	N/ui/serverWidget Module	
nlobjAssistant.addField(name, type, label, source, group)	Assistant.addField(options)	N/ui/serverWidget Module	
nlobjAssistant.addFieldGroup(name, label)	Assistant.addFieldGroup(options)	N/ui/serverWidget Module	
nlobjAssistant.addStep(name, label)	Assistant.addStep(options)	N/ui/serverWidget Module	
nlobjAssistant.addSubList(name, type, label)	Assistant.addSublist(options)	N/ui/serverWidget Module	
nlobjAssistant.getAllFields()	Assistant.getFieldIds()	N/ui/serverWidget Module	
nlobjAssistant.getAllFieldGroups()	Assistant.getFieldGroupIds()	N/ui/serverWidget Module	
nlobjAssistant.getAllSteps()	Assistant.getSteps()	N/ui/serverWidget Module	
nlobjAssistant.getAllSubLists()	Assistant.getSublistIds()	N/ui/serverWidget Module	
nlobjAssistant.getCurrentStep()	Assistant.currentStep	N/ui/serverWidget Module	Note that <code>currentStep</code> is a property.
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	

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjAssistant.getSubList(name)	Assistant.getList(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 <code>currentStep</code> is a property.
nlobjAssistant.setError(html)	Assistant.errorHtml	N/ui/serverWidget Module	Note that <code>errorHtml</code> is a property.
nlobjAssistant.setFieldValues(values)	Assistant.updateDefaultValues(values)	N/ui/serverWidget Module	
nlobjAssistant.setFinished(html)	Assistant.finishedHtml	N/ui/serverWidget Module	Note that <code>finishedHtml</code> is a property.
nlobjAssistant.setNumbered(hasStepNumber)	Assistant.hideStepNumber	N/ui/serverWidget Module	Note that <code>hideStepNumber</code> is a property.
nlobjAssistant.setOrdered(order)	Assistant.isNotOrdered	N/ui/serverWidget Module	Note that <code>isNotOrdered</code> is a property.
nlobjAssistant.setScript(script)	Assistant.clientScriptFileId Assistant.clientScriptModulePath	N/ui/serverWidget Module	Note that <code>clientScriptFileId</code> and <code>clientScriptModulePath</code> 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 <code>hideAddToShortcutsLink</code> 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 <code>title</code> 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.getAllLineItemGroup()	AssistantStep.getSublistFieldIds(options)	N/ui/serverWidget Module	
nlobjAssistantStep.getAllLineItems()	AssistantStep.getSubmittedSublistIds()	N/ui/serverWidget Module	
nlobjAssistantStep.getFieldValue(options)	AssistantStep.getValue(options)	N/ui/serverWidget Module	

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjAssistantStep.getFieldValues(options)	AssistantStep.getValue(options)	N/ui/serverWidget Module	
nlobjAssistantStep.getLineItem group)	AssistantStep. getLineCount(options)	N/ui/serverWidget Module	
nlobjAssistantStep.getLineItem group, name, line)	AssistantStep. getSublistValue(options)	N/ui/serverWidget Module	
nlobjAssistantStep.getStepNum ber	AssistantStep. stepNumber	N/ui/serverWidget Module	Note that stepNumber is a property.
nlobjAssistantStep.setHelpText(helpText)	AssistantStep.helpText	N/ui/serverWidget Module	Note that helpText is a property.
nlobjAssistantStep.setLabel(label)	AssistantStep.label	N/ui/serverWidget Module	Note that label 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(disable d)	Button.isEnabled	N/ui/serverWidget Module	Note that isEnabled is a property.
nlobjButton.setLabel(label)	Button.label	N/ui/serverWidget Module	Note that label is a property.
nlobjButton.setVisible(visible)	Button.isHidden	N/ui/serverWidget Module	Note that isHidden is a property.

nlobjColumn

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjColumn	serverWidget.ListColumn	N/ui/serverWidget Module	
nlobjColumn.addParamToURL(value, dynamic)	ListColumn. addParamToURL(options)	N/ui/serverWidget Module	
nlobjColumn.setLabel(label)	ListColumn.label	N/ui/serverWidget Module	Note that label 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 N/config Module 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 N/config Module Script Sample .
nlobjConfiguration.getAllFields()	Record.getFields()	N/record Module	
nlobjConfiguration.getField(fldName)	Record.getField(options)	N/record Module	
nlobjConfiguration.getFieldText(recordId)	Record.getText(options)	N/record Module	
nlobjConfiguration.getFieldTexts(recordId)	Record.getText(options)	N/record Module	
nlobjConfiguration.getFieldValue(recordId)	Record.getValue(options)	N/record Module	
nlobjConfiguration.getFieldValue(recordId)	Record.getValue(options)	N/record Module	
nlobjConfiguration.getType()	Record.type	N/record Module	Note that <code>type</code> is a property.
nlobjConfiguration.setTextText(recordId)	Record.setText(options)	N/record Module	
nlobjConfiguration.setTextTexts(recordId)	Record.setText(options)	N/record Module	
nlobjConfiguration.setValue(recordId)	Record.setValue(options)	N/record Module	
nlobjConfiguration.setValue(recordId)	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.getCompany()	runtime.accountId	N/runtime Module	Note that <code>accountId</code> is a property.
nlobjContext.getDepartment()	User.department	N/runtime Module	Note that <code>department</code> is a property.
nlobjContext.getDeploymentId()	Script.deploymentId	N/runtime Module	Note that <code>deploymentId</code> is a property.
nlobjContext.getEmail()	User.email	N/runtime Module	Note that <code>email</code> is a property.
nlobjContext.getEnvironment()	runtime.envType	N/runtime Module	Note that <code>envType</code> is a property.

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjContext.getExecutionContext()	runtime.executionContext	N/runtime Module	Note that <code>executionContext</code> is a property.
nlobjContext.getFeature(name)	runtime.isFeatureInEffect(options)	N/runtime Module	
nlobjContext.getLocation()	User.location	N/runtime Module	Note that <code>location</code> is a property.
nlobjContext.getLogLevel()	Script.logLevel	N/runtime Module	Note that <code>logLevel</code> is a property.
nlobjContext.getName()	User.name	N/runtime Module	Note that <code>name</code> is a property.
nlobjContext.getPercentComplete()	Script.percentComplete	N/runtime Module	Note that <code>percentComplete</code> is a property. For a script sample, see N/runtime Module Script Sample.
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 <code>queueCount</code> is a property.
nlobjContext.getRemainingUsage()	Script.getRemainingUsage()	N/runtime Module	
nlobjContext.getRole()	User.role	N/runtime Module	Note that <code>role</code> is a property.
nlobjContext.getRoleCenter()	User.roleCenter	N/runtime Module	Note that <code>roleCenter</code> is a property.
nlobjContext.getRoleId()	User.roleId	N/runtime Module	Note that <code>roleId</code> is a property.
nlobjContext.getScriptId()	Script.id	N/runtime Module	Note that <code>id</code> 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 <code>Script.getParameter(options)</code> is equivalent to <code>nlobjContext.getSetting('SCRIPT', name)</code> . The method <code>Session.get(options)</code> is equivalent to <code>nlobjContext.getSetting('SESSION', name)</code> . The method <code>runtime.isFeatureInEffect(options)</code> is equivalent to <code>nlobjContext.getSetting('FEATURE', name)</code> The method <code>User.getPermission(options)</code> is equivalent to <code>nlobjContext.getSetting('PERMISSION', name)</code>

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjContext.getSubsidiary()	User.subsidiary	N/runtime Module	Note that <code>subsidiary</code> is a property.
nlobjContext.getUser()	User.id	N/runtime Module	Note that <code>id</code> is a property.
nlobjContext.getVersion()	runtime.version	N/runtime Module	Note that <code>version</code> is a property.
nlobjContext.setPercentComplete(pct)	Session.percentComplete	N/runtime Module	Note that <code>percentComplete</code> 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	
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(<i>linkedFile</i>)	CsvImportTask.linkedFiles	N/task Module	Note that <code>linkedFiles</code> is a property.
nlobjCSVImport.setMapping(<i>savedImport</i>)	CsvImportTask.mappingId	N/task Module	Note that <code>mappingId</code> is a property.
nlobjCSVImport.setOption(<i>optionValue</i>)	CsvImportTask.name	N/task Module	Note that <code>name</code> is a property.
nlobjCSVImport.setPrimaryFile(<i>importFile</i>)	CsvImportTask.importFile	N/task Module	Note that <code>importFile</code> is a property.
nlobjCSVImport.setQueue(<i>queueId</i>)	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(<i>entityType</i>)	EntityDeduplicationTask.entityType	N/task Module	Note that <code>entityType</code> is a property.
nlobjDuplicateJobRequest.setMasterId(<i>masterID</i>)	EntityDeduplicationTask.masterRecordId	N/task Module	
nlobjDuplicateJobRequest.setMasterSelectionMode(<i>mode</i>)	EntityDeduplicationTask.masterSelectionMode	N/task Module	Note that <code>masterSelectionMode</code> is a property.
nlobjDuplicateJobRequest.setOperation(<i>operation</i>)	EntityDeduplicationTask.dedupeMode	N/task Module	Note that <code>dedupeMode</code> is a property.
nlobjDuplicateJobRequest.setRecords(<i>dupeRecords</i>)	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()		N/render Module	
nlobjEmailMerger.setCustomRecordType(<i>recordId</i>)	See Notes	N/render Module	In SuiteScript 2.0, this value is set with a <code>render.mergeEmail(options)</code> parameter.
nlobjEmailMerger.setEntity(<i>entityId</i>)	See Notes	N/render Module	In SuiteScript 2.0, this value is set with a

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
			render.mergeEmail(options) parameter.
nlobjEmailMerger.setRecipient(recipientType, recipientId)	See Notes	N/render Module	In SuiteScript 2.0, this value is set with a render.mergeEmail(options) parameter.
nlobjEmailMerger.setSupportCaseId(caseld)	See Notes	N/render Module	In SuiteScript 2.0, this value is set with a render.mergeEmail(options) parameter.
nlobjEmailMerger.setTransactionId(transactionId)	See Notes	N/render Module	In SuiteScript 2.0, this value is set with a render.mergeEmail(options) 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()		N/error Module	
nlobjError.getDetails()		N/error Module	
nlobjError.getId()		N/error Module	
nlobjError.getInternalId()		N/error Module	
nlobjError.getStackTrace()		N/error Module	
nlobjError.getUserEvent()		N/error Module	

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 N/ui/ serverWidget module to create and modify form fields in a Suitelet. Use the N/record 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	

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjFieldGroup.setCollapsible(collapsible, hidden)	FieldGroup.isCollapsible	N/ui/serverWidget Module	Note that isCollapsible is a property.
nlobjFieldGroup.setLabel(label)	FieldGroup.label	N/ui/serverWidget Module	Note that label is a property.
nlobjFieldGroup.setShowBorder(borderHidden)	FieldGroup. isBorderHidden	N/ui/serverWidget Module	Note that isBorderHidden is a property.
nlobjFieldGroup.setSingleColumn(column)	FieldGroup. isSingleColumn	N/ui/serverWidget Module	Note that isSingleColumn 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 description is a property.
nlobjFile.getFolder()	File.folder	N/file Module	Note that folder is a property. For a script sample, see N/file Module Script Sample .
nlobjFile.getId()	File.id	N/file Module	Note that id is a property. For a script sample, see N/file Module Script Sample .
nlobjFile.getName()	File.name	N/file Module	Note that name is a property. For a script sample, see N/file Module Script Sample .
nlobjFile.getSize()	File.size	N/file Module	Note that size is a property.
nlobjFile.getType()	File.fileType	N/file Module	Note that fileType is a property. For a script sample, see N/file Module Script Sample . For a script sample, see N/file Module Script Sample .
nlobjFile.getURL()	File.url	N/file Module	Note that url is a property.
nlobjFile.getValue()	File.getContents()	N/file Module	
nlobjFile.isInactive()	File.isInactive	N/file Module	Note that isInactive is a property.
nlobjFile.isOnline()	File.isOnline	N/file Module	Note that isOnline 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 .
nlobjFile.setDescription(description)	File.description	N/file Module	Note that <code>description</code> is a property.
nlobjFile.setEncoding(encodingType)	File.encoding	N/file Module	Note that <code>encoding</code> is a property.
nlobjFile.setFolder(id)	File.folder	N/file Module	Note that <code>folder</code> is a property. You can also set the folder during file creation with <code>file.create(options)</code> . For a script sample, see N/file Module Script Sample .
nlobjFile.setIsInactive(inactive)	File.isInactive	N/file Module	Note that <code>isInactive</code> is a property.
nlobjFile.setIsOnline(online)	File.isOnline	N/file Module	Note that <code>isOnline</code> is a property. For a script sample, see N/file Module Script Sample .
nlobjFile.setName(name)	File.name	N/file Module	Note that <code>name</code> 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(name, label, script)	Form.addButton(options)	N/ui/serverWidget Module	
nlobjForm.addCredentialField(id, label, website, scriptId, value, entityMatch, tab)	Form.addCredentialField(options)	N/ui/serverWidget Module	
nlobjForm.addField(name, type, label, sourceOrRadio, tab)	Form.addField(options)	N/ui/serverWidget Module	For a script sample, see N/ui/serverWidget Module Script Sample
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 N/ui/serverWidget Module Script Sample
nlobjForm.addButton(label)	Form.addSubmitButton(options)	N/ui/serverWidget Module	For a script sample, see N/ui/serverWidget Module Script Sample

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjForm.addSubTab(name, label, tab)	Form.addTab(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.clientScriptFileId Form.clientScriptModulePath	N/ui/serverWidget Module	Note that clientScriptFileId and clientScriptModulePath are properties. Use one of these SuiteScript 2.0 properties to attach an ad hoc client script to a form.
nlobjForm.setTitle(title)	Form.title	N/ui/serverWidget Module	Note that title 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 status is a property.
nlobjFuture.isDone()	EntityDeduplicationTaskStatus.status	N/task Module	Note that status is a property.

nlobjJobManager

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjJobManager		N/task Module	
nlobjJobManager.createJobRequest()		N/task Module	
nlobjJobManager.getFuture()		N/task Module	
nlobjJobManager.submit(nlobjDuplicatedJobRequest)		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	
nlobjList.setScript(script)	List.clientScriptFileDialog List.clientScriptModulePath	N/ui/serverWidget Module	Note that clientScriptFileDialog and clientScriptModulePath 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 style is a property.
nlobjList.setTitle(title)	List.title	N/ui/serverWidget Module	Note that title 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/

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
			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 Portlet Script Type .
nlobjPortlet.addColumn(name, type, label, just)	Portlet.addColumn(options)	N/A	For additional information, see Portlet Script Type .
nlobjPortlet.addEditColumn(column, showView, showHrefCol)	Portlet.addEditColumn(options)	N/A	For additional information, see Portlet Script Type .
nlobjPortlet.addField(name, type, label, source)	Portlet.addField(options)	N/A	For additional information, see Portlet Script Type .
nlobjPortlet.addLine(text, url, indent)	Portlet.addLine(options)	N/A	For additional information, see Portlet Script Type .
nlobjPortlet.addRow(row)	Portlet.addRow(options)	N/A	For additional information, see Portlet Script Type .
nlobjPortlet.addRows(rows)	Portlet.addRows(options)	N/A	For additional information, see Portlet Script Type .

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjPortlet.setHtml(html)	Portlet.html	N/A	For additional information, see Portlet Script Type .
nlobjPortlet.setRefreshInterval(n)		N/A	For additional information, see Portlet Script Type .
nlobjPortlet.setScript(scriptid)	Portlet.clientScriptFileId Portlet.clientScriptModulePath	N/A	For additional information, see Portlet Script Type .
nlobjPortlet.setSubmitButton(url, label, target)	Portlet.setSubmitButton(options)	N/A	For additional information, see Portlet Script Type .
nlobjPortlet.setTitle(title)	Portlet.title	N/A	For additional information, see Portlet Script Type .

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 Scripting Subrecords .
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 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
			SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under Scripting Subrecords .
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 Scripting Subrecords .
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 Scripting Subrecords .
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

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
			this functionality in SuiteScript 2.0.
nlobjRecord.getCurrentLineItemMatrixValue(group, fldnam, column)	Record.getCurrentMatrixSublistValue(options) CurrentRecord.getCurrentMatrixSublistValue(options)	N/record Module N/currentRecord Module	
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.getDate TValue(fieldId, timeZone)	See Notes	N/format Module	Use the N/format 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.getLineItemDate TValue(type, fieldId, lineNumber, timeZone)	See Notes	N/format Module	Use the N/format module to mimic this functionality in SuiteScript 2.0.
nlobjRecord.getLineItemField(group, fldnam, linenum)	Record.getSublistField(options)	N/record Module	

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
	CurrentRecord. getSublistField(options)	N/currentRecord Module	
nlobjRecord.getLineItemMatrix group, fldnam, linenum, column)	Record.getMatrixSublistF ield(options) CurrentRecord. getMatrixSublistField(options)	N/record Module N/currentRecord Module	
nlobjRecord.getLineItemM atrixValue(group, fldnam, lineum, column)	Record.getMatrixSublist Value(options) CurrentRecord. getMatrixSublistValue(options)	N/record Module N/currentRecord Module	
nlobjRecord.getLineItemText(gro fldnam, linenum)	Record.getSublistText(options) CurrentRecord. getSublistText(options)	N/record Module N/currentRecord Module	
nlobjRecord.getLineItemValue(group, name, linenum)	Record.getSublistValue(options) CurrentRecord. getSublistValue(options)	N/record Module N/currentRecord Module	
nlobjRecord.getLineItemValues(type, fldnam, linenum)	Record.getSublistValue(options) CurrentRecord. getSublistValue(options)	N/record Module N/currentRecord Module	
nlobjRecord.getMatrixCount(gro fldnam)	Record.getMatrixHeader Count(options) CurrentRecord. getMatrixHeaderCount(options)	N/record Module N/currentRecord Module	
nlobjRecord.getMatrixField(gro fldname, column)	Record.getMatrixHeader Field(options) CurrentRecord. getMatrixHeaderField(options)	N/record Module N/currentRecord Module	
nlobjRecord.getMatrixValue(gro fldnam, column)	Record.getMatrixHeader Value(options) CurrentRecord. getMatrixHeaderValue(options)	N/record Module N/currentRecord Module	
nlobjRecord.getRecordType()	Record.type	N/record Module	
nlobjRecord.insertLineItem(gro linenum, ignoreRecalc)	Record.insertLine(options) CurrentRecord. insertLine(options)	N/record Module N/currentRecord Module	
nlobjRecord.removeLineItem(gro linenum, ignoreRecalc)	Record.removeLine(options) CurrentRecord. removeLine(options)	N/record Module N/currentRecord Module	

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
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 Scripting Subrecords .
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 Scripting Subrecords .
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 N/format 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	
nlobjRecord.setCurrentLineItemValue(group, name, value)	Record.setCurrentSublistValue(options) CurrentRecord.setCurrentSublistValue(options)	N/record Module N/currentRecord Module	

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjRecord.setDateTimeValue(fieldId, dateTime, timeZone)	See Notes	N/format Module	Use the N/format 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.setFieldValues(name, value)	Record.setValue(options) CurrentRecord.setValue(options)	N/record Module N/currentRecord Module	
nlobjRecord.setLineItem dateTimeValue(type, fieldId, lineNum, dateTime, timeZone)	See Notes	N/format Module	Use the N/format 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.setMatrixHeader Value(options) CurrentRecord. setMatrixHeaderValue(options)	N/record Module N/currentRecord Module	
nlobjRecord.viewCurrentL ineItemSubrecord(sublist, fldname)	Record.getCurrentSublis tSubrecord(options) CurrentRecord. getCurrentSublistSubrec ord(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 Scripting Subrecords .
nlobjRecord.viewLineItemS ubrecord(sublist, fldname, linenum)	Record.getSublistSubrec ord(options)	N/record Module	Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
			SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under Scripting Subrecords .
nlobjRecord.viewSubrecord(fieldName)	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 Scripting Subrecords .

nlobjRequest

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjRequest	http.ServerRequest		■ test ■ test
nlobjRequest.getAllHeaders()			
nlobjRequest.getAllParameters()			
nlobjRequest.getBody()			
nlobjRequest.getFile(id)			
nlobjRequest.getHeader(name)			
nlobjRequest.getLineItemCount(group)			
nlobjRequest.getLineItemValue(group, name, line)			
nlobjRequest.getMethod()			
nlobjRequest.getParameter(name)			
nlobjRequest.getParameterValues(name)			
nlobjRequest.getURL()			

nlobjResponse

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjResponse			
nlobjResponse.addHeader(name, value)			
nlobjResponse.getAllHeaders()			
nlobjResponse.getBody()			
nlobjResponse			
nlobjResponse.getCode()			
nlobjResponse.getError()			
nlobjResponse.getHeader(name)			
nlobjResponse.getHeaders(name)			
nlobjResponse.renderPDF(xmlString)			
nlobjResponse.setCDNCacheable(type)			
nlobjResponse.setContentType(type, name, disposition)			
nlobjResponse.setEncoding(encodingType)			
nlobjResponse.setHeader(name, value)			
nlobjResponse.sendRedirect(type, identifier, id, editmode, parameters)			
nlobjResponse.write(output)			
nlobjResponse.writeLine(output)			
nlobjResponse.writePage(pageobject)			

nlobjSearch

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjSearch	search.Search	N/search Module	
nlobjSearch.addColumn(column)		N/search Module	
nlobjSearch.setColumns(columns)		N/search Module	
nlobjSearch.addFilter(filter)		N/search Module	
nlobjSearch.addFilters(filters)		N/search Module	
nlobjSearch.deleteSearch()	search.delete(options)	N/search Module	For a script sample, see N/search Module Script Samples .

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
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.
nlobjSearch.getId()	Search.searchId	N/search Module	Note that <code>id</code> is a property.
nlobjSearch.getIsPublic()	Search.isPublic	N/search Module	Note that <code>isPublic</code> is a property.
nlobjSearch.getScriptId()	Search.id	N/search Module	
nlobjSearch.getSearchType()	Search.searchType	N/search Module	Note that <code>searchType</code> 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 <code>columns</code> is a property.
nlobjSearch.setFilterExpression(filterExpression)	Search.filterExpression	N/search Module	Note that <code>filterExpression</code> is a property.
nlobjSearch.setFilters(filters)	Search.filters	N/search Module	Note that <code>filters</code> is a property.
nlobjSearch.setIsPublic(type)	Search.isPublic	N/search Module	Note that <code>isPublic</code> 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 nlobjSelectOption methods.
nlobjSelectOption.getId()	N/record: Field.getSelectOptions(options) N/currentRecord: Field.getSelectOptions(options)	N/record Module N/currentRecord Module	
nlobjSelectOption.getText()	N/record: Field.getSelectOptions(options) N/currentRecord: Field.getSelectOptions(options)	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	

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjSublist.addRefreshButton()	Sublist.addRefreshButton()	N/ui/serverWidget Module	
nlobjSublist.getLineItemCount()	Sublist.lineCount	N/ui/serverWidget Module	Note that lineCount is a property
nlobjSublist.getLineItemValue(group, fldnam, linenum)	Sublist.getSublistValue(options)	N/ui/serverWidget Module	
nlobjSublist.setAmountField(field)		N/ui/serverWidget Module	
nlobjSublist.setDisplayType(type)	Sublist.displayType	N/ui/serverWidget Module	Note that displayType is a property
nlobjSublist.setHelpText(help)	Sublist.helpText	N/ui/serverWidget Module	Note that helpText is a property
nlobjSublist.setLabel(label)	Sublist.label	N/ui/serverWidget Module	Note that label is a property
nlobjSublist.setLineItemValue(name, linenum,value)	Sublist.setSublistValue(options)	N/ui/serverWidget Module	
nlobjSublist.setLineItemValues(values)		N/ui/serverWidget Module	
nlobjSublist.setUniqueField(name)	Sublist.updateUniqueId(options)	N/ui/serverWidget Module	Note that uniqueFieldId is a property

nlobjSubrecord

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjSubrecord	See Notes	N/record Module	SuiteScript 2.0 subrecords are returned as record.Record objects. Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under Scripting Subrecords.
nlobjSubrecord.cancel()	See Notes	N/record Module	SuiteScript 2.0 subrecords are returned as record.Record objects. Note that scripting subrecords in SuiteScript 2.0 is fundamentally different

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
			from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under Scripting Subrecords .
nlobjSubrecord.commit()	See Notes	N/record Module	This API does not have a SuiteScript 2.0 equivalent. SuiteScript 2.0 subrecords are returned as record.Record objects. Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under Scripting Subrecords .

nlobjTab

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

nlobjTemplateRenderer

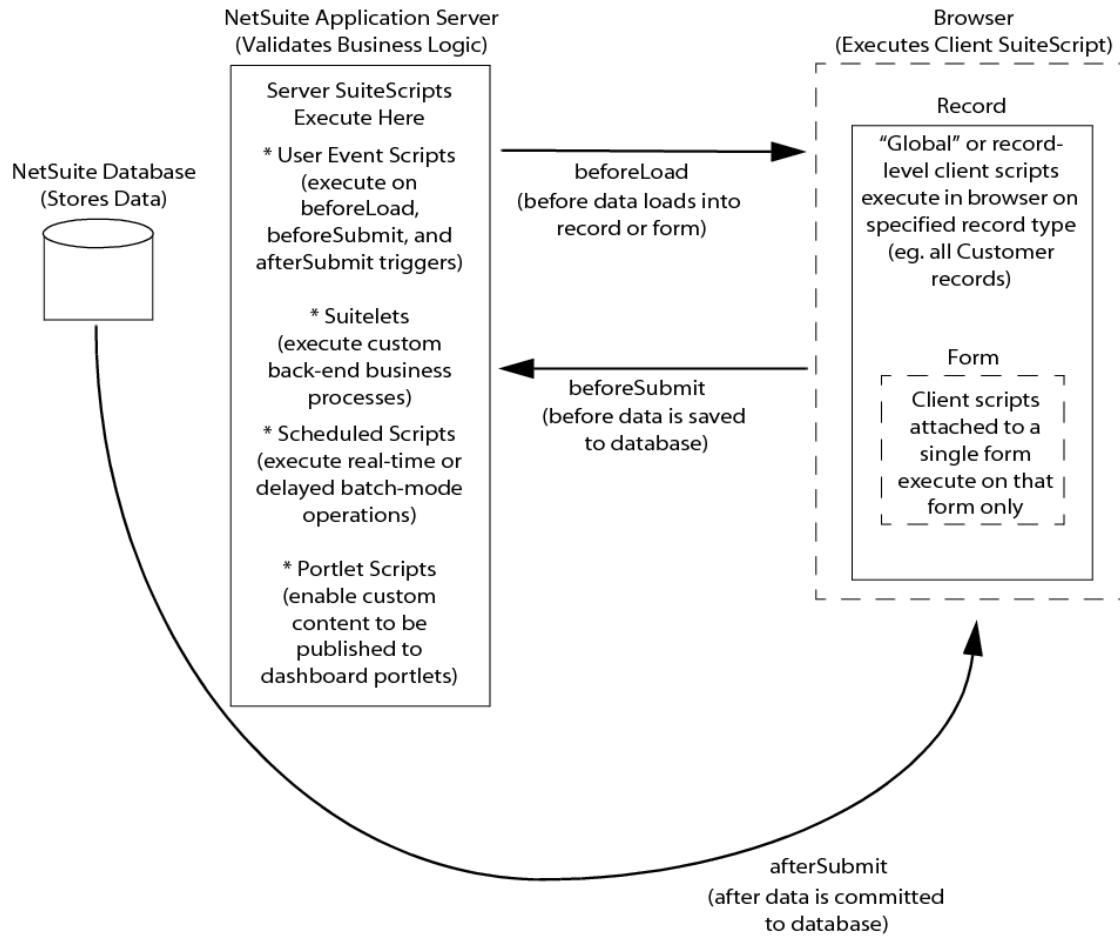
SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjTemplateRenderer	render.TemplateRenderer	N/render Module	For a script sample, see N/render Module Script Sample .
nlobjTemplateRenderer.addRecord(record)	TemplateRenderer.addRecord(options)	N/render Module	For a script sample, see N/render Module Script Sample .
nlobjTemplateRenderer.addSearchResult(searchResult)	TemplateRenderer.addSearchResults(options)	N/render Module	For a script sample, see N/render Module Script Sample .

SuiteScript 1.0 API	SuiteScript 2.0 API	SuiteScript 2.0 Module	Notes
nlobjTemplateRenderer.renderToResponse()	TemplateRenderer.renderToResponse(options)	N/render Module	
nlobjTemplateRenderer.renderToString()	TemplateRenderer.renderAsString()	N/render Module	
nlobjTemplateRenderer.setTemplateContent()	TemplateRenderer.setTemplateContent	N/render Module	For a script sample, see N/render Module Script Sample .

SuiteScript 2.0 Script Types and Entry Points

To understand SuiteScript script types and how to debug SuiteScript code, it is important to understand **where** scripts run (client-side or server-side) and **when** they run (before data loads into a page loads, after an update is made to the data, or after the data has been saved and committed to the database).

The following diagram shows where and when script types run:



SuiteScript 2.0 Script Types

The following script types are supported:

- [Bundle Installation Script Type](#)

Bundle installation scripts fire triggers that execute as part of bundle installation, update, or uninstall. Trigger execution can occur either before install, after install, before update, after update, or before uninstall. These triggers automatically complete required setup, configuration, and data management tasks for the bundle.

- [Client Script Type](#)

Client scripts run on individual forms, can be deployed globally, and are applied to entity and transaction record types. Global client scripts enable centralized management of scripts that can be applied to an entire record type.

- [Map/Reduce Script Type](#)

Map/reduce scripts update records as part of a mass update.

- [Mass Update Script Type](#)

Mass update scripts allows you to programmatically perform custom mass updates to update fields that are not available through general mass updates. These scripts can run complex calculations across many records.

- [Portlet Script Type](#)

Portlet scripts create custom dashboard portlets. For example, you can use SuiteScript to create a portlet that is populated on-the-fly with company messages based on data within the system.

- [RESTlet Script Type](#)

RESTlets are server-side scripts that can be used to define custom RESTful integrations to NetSuite.

- [Scheduled Script Type](#)

Scheduled scripts are executed on-demand in real-time or via a user-configurable schedule. These scripts perform batch processing of records.

- [Suitelet Script Type](#)

Suitelets enable the creation of dynamic web content and build NetSuite-looking pages. Suitelets can be used to implement custom front and backends.

- [User Event Script Type](#)

User Event scripts are triggered when users work with records and data changes in NetSuite as they create, open, update, or save records. These scripts customize the workflow and association between your NetSuite entry forms. These scripts can also be used for doing additional processing before records are entered or for validating entries based on other data in the system.

- [Workflow Action Script Type](#)

Workflow action scripts allow you to create custom actions that are defined on a record in a workflow.

 **Note:** For information on creating a SuiteScript 2.0 Entry Point Script, see [SuiteScript 2.0 Entry Point Script Deployment](#).

Best practices

- Always thoroughly test your code before using it on your live NetSuite data.
- Type all record, field, sublist, tab, and subtab IDs in lowercase in your SuiteScript code.
- Prefix all custom script IDs and deployment IDs with an underscore (_).
- Do not hard-code any passwords in scripts. The password and password2 fields are supported for scripting.
- If the same code is used across multiple forms, ensure that you test any changes in the code for each form that the code is associated with.
- Include proper error handling sequences in your script wherever data may be inconsistent, not available, or invalid for certain functions. For example, if your script requires a field value to validate another, ensure that the field value is available.
- Organize your code into reusable chunks. Many functions can be used in a variety of forms. Any reusable functions should be stored in a common library file and then called into specific event functions for the required forms as needed.

- Place all custom code and markup, including third party libraries, in your own namespace.

Important: Custom code must not be used to access the NetSuite DOM. Developers must use SuiteScript APIs to access NetSuite UI components.

- Use the built in Library functions whenever possible for reading/writing Date/Currency fields and for querying XML documents
- During script development, componentize your scripts, load them individually, and then test each one — inactivating all but the one you are testing when multiple components are tied to a single user event.
- When working with script type events, your function name should correspond with the event. For example, a pageInit event can be named PagelInit or formAPageInit.
- Since name values can change, ensure that you use **static** ID values in your API calls where applicable.
- Although you can use any desired naming conventions for functions within your code, it is recommended that you use custom namespaces or unique prefixes for all your function names.
- Thoroughly comment your code. This practice helps with debugging and development and assists NetSuite support in locating problems if necessary.
- You must use the `runtime.getCurrentScript()` function in the runtime module to reference script parameters. For example, use the following code to obtain the value of a script parameter named `custscript_case_field`:

```
define(['N/runtime'], function(runtime) {
    function pageInit(context) {
        var strField = runtime.getCurrentScript().getParameter('SCRIPT', 'custscript_case_field');
    }
});
```

Bundle Installation Script Type

Bundle installation scripts are specialized server scripts that perform processes in target accounts as part of a bundle installation, update, or uninstall. These processes include setup, configuration, and data management tasks that would otherwise have to be completed by account administrators.

Every bundle can include a bundle installation script that is automatically run when the bundle is installed, upgraded, or uninstalled. Each bundle installation script can contain triggers to be executed before install, after install, before update, after update, and after uninstall.

Bundle installation script failures terminate bundle installations, updates, or uninstalls. Bundle installation scripts can include their own error handling in addition to errors thrown by SuiteBundler and the SuiteScript engine. An error thrown by a bundle installation script returns an error code of **Installation Error** followed by the text defined by the script author.

Bundle Installation Script Sample

The script sample performs the following tasks:

- Before the bundle installation and the bundle update, ensure that the Time Off Management feature is enabled in the target NetSuite account and warn the user if the feature is not enabled.

```
/**
```

```

* @NApiVersion 2.0
* @NScriptType BundleInstallationScript
*/
define(['N/runtime'], function(runtime) {
    function checkPrerequisites() {
        if (!runtime.isFeatureInEffect({
            feature: 'TIMEOFFMANAGEMENT'
        })) {
            throw 'The TIMEOFFMANAGEMENT feature must be enabled. ' +
                'Please enable the feature and try again.';
        }
        return {
            beforeInstall: function beforeInstall(params) {
                checkPrerequisites();
            },
            beforeUpdate: function beforeUpdate(params) {
                checkPrerequisites();
            }
        };
    });
});

```

Bundle Installation Script Entry Points

Script Entry Point	
afterInstall	Executes after a bundle is installed for the first time in a target account.
afterUpdate	Executes after a bundle in a target account is updated.
beforeInstall	Executes before a bundle is installed for the first time in a target account.
beforeUninstall	Executes before a bundle is uninstalled from a target account.
beforeUpdate	Executes before a bundle in a target account is updated.

afterInstall

Description	Executes after a bundle is installed for the first time in a target account.
Returns	void
Since	Version 2016 Release 1

Parameters

Note: The `params` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
<code>params.version</code>	number	The version of the bundle that is being installed in the target account.	Version 2016 Release 1

afterUpdate

Description	Executes after a bundle in a target account is updated.
Returns	void
Since	Version 2016 Release 1

Parameters

Note: The `params` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
<code>params.fromVersion</code>	number	The version of the bundle that is currently installed in the target account.	Version 2016 Release 1
<code>params.toVersion</code>	number	The version of the bundle that is being installed in the target account.	Version 2016 Release 1

beforeInstall

Description	Executes before a bundle is installed for the first time in a target account. Calls to scheduled scripts are not supported.
Returns	void
Since	Version 2016 Release 1

Parameters

Note: The `params` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
<code>params.version</code>	number	The version of the bundle that is being installed in the target account.	Version 2016 Release 1

beforeUninstall

Description	Executes before a bundle is uninstalled from a target account.
-------------	--

	Calls to scheduled scripts are not supported.
Returns	void
Since	Version 2016 Release 1

Parameters

i Note: The `params` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
<code>params.version</code>	number	The version of the bundle that is being uninstalled from the target account.	Version 2016 Release 1

beforeUpdate

Description	Executes before a bundle in a target account is updated. Calls to scheduled scripts are not supported.
Returns	void
Since	Version 2016 Release 1

Parameters

i Note: The `params` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
<code>params.fromVersion</code>	number	The version of the bundle that is currently installed in the target account.	Version 2016 Release 1
<code>params.toVersion</code>	number	The version of the bundle that is being installed in the target account.	Version 2016 Release 1

Client Script Type

Client scripts are scripts that are executed by predefined event triggers in a browser. They can validate user-entered data and auto-populate fields or sublists at various form events.

Scripts can be run on most standard records, custom record types, and custom NetSuite pages such as Suitelets.

The following triggers can run a client script.

- Initializing a form
- Entering or changing a value in a field (before and after it is entered)
- Entering or changing a value in a field that sources another field
- Selecting a line item on a sublist
- Adding a line item (before and after it is entered)
- Saving a form

Record-level client scripts are executed after any existing form-based clients are run, and before any user event scripts are run.

- [Remote Objects in Client Scripts](#)
- [Role Restrictions](#)
- [Best Practices](#)
- [Client Script Sample](#)

Remote Objects in Client Scripts

A client script interfaces with remote objects whenever it calls the NetSuite database to create, load, copy, or transform an object record.

The following is a sample client script. When the saveRecord client event is triggered, the script uses the currentRecord module to validate a Sales Order record. The script ensures that the transaction date is not older than one week, and that the total is valid.

```
/** 
 * @NApiVersion 2.x
 * @NScriptType ClientScript
 */
define(['N/ui/message'],
    function(msg) {
        function showErrorMessage(msgText) {
            var myMsg = msg.create({
                title: "Cannot Save Record",
                message: msgText,
                type: msg.Type.ERROR
            });

            myMsg.show({
                duration: 5000
            });
        }

        function saveRec(context) {
            var rec = context.currentRecord;
            var currentDate = new Date()
            var oneWeekAgo = new Date(currentDate - 1000 * 60 * 60 * 24 * 7);

            // Validate transaction date is not older than current time by one week
            if (rec.getValue({
```

```

        fieldId: 'trandate'
    }) < oneWeekAgo) {
    showErrorMessage("Cannot save sales order with trandate one week old.");
    return false;
}

//validate total is greater than 0
if (rec.getValue({
    fieldId: 'total'
}) <= 0) {
    showErrorMessage("Cannot save sales order with negative total amount.");
    return false;
}
return true;
}

return {
    saveRecord: saveRec
}
});

```

For more information about the currentRecord module, see [N/currentRecord Module](#).

Role Restrictions

Client scripts respect the role permissions specified in the user's NetSuite account. An error is thrown when running a client script to access a record with a role that does not have permission to view or edit the record.

The following client script attaches to a custom sales order form and executes on the field change client event:

```

/**
 * @NApiVersion 2.x
 * @NScriptType ClientScript
 */
define(['N/search'],
    function(search) {
        function getSalesRepEmail(context) {
            var salesRep = context.currentRecord.getValue({
                fieldId: 'salesrep'
            });
            var salesRepEmail = search.lookupFields({
                type: 'employee',
                id: salesRep,
                columns: ['email']
            });
            alert(JSON.stringify(salesRepEmail));
        }

        return {
            fieldChanged: getSalesRepEmail
        }
});

```

If you are logged in with an admin role, you receive the alert when you load the sales order with this form. If you are logged in with a role that does not have permission to view/edit Employee records, you receive an error when you select the Sales Rep field.

The following considerations can help prevent users from receiving the error:

- Consider the types of users who may be using your custom form and running the script.
- Consider which record types that users do not have access to. If it is vital that all who run the script have access to the records in the script, you may need to redefine the permissions of the users (if your role is as an admin).
- Consider rewriting your script so that it only references record types that all users have access to.
- Consider writing the script as a server-side user event script, and set the Execute As Admin preference on the Script Deployment page. Note that alerts are a function of client scripts only and cannot be used in user event scripts. For more information about user event scripts, see [User Event Script Type](#).

Best Practices

The following list shows best practices for both form-level and record-level client script development:

- When testing form-level client scripts, ensure that the latest scripts are being executed. Use Ctrl-Refresh to clear your browser cache.
- Use global (record-level) client scripts to get a more flexible deployment model and to port (bundle) scripts.
- `record.setValue` and `record.setCurrentSublistValue` execution is multi-threaded whenever child field values need to be sourced in. To synchronize your logic, use the `postSourcing` function or set the `fireSlavingSync` synchronous parameter to true.
- Methods that set values accept raw data of a specific type and do not require formatting or parsing. Methods that set text accept strings but can conform to a user-specified format. For example, when setting a numerical field type, `setValue` accepts any number, and `setText` accepts a string in a specified format. (such as "2,000.39" or "2.00,39") When setting a Date field type, `setValue` accepts any Date object and `setText` accepts a string in a specified format. (such as "6/9/2016" or "9/6/2016")

Client Script Sample

The following sample shows a client script applied to a sales order form. The script performs the following tasks:

- When the form loads for editing, the `pageInit` event sets the customer field to a specific customer.
- When form changes are saved, the `saveRecord` event ensures that the customer field is set and at least one sales order item is listed.
- When editing the quantity of a sales order item, the `validateField` event ensures that the number is less than three.
- When selecting a sales order item, the `fieldChanged` event updates a memo field to indicate that the item was selected.
- When a specific sales order item is selected, the `postSourcing` event updates the price level for that particular item.
- When an existing partner is selected, the `initLine` event changes the selected partner to a specific partner.
- When a partner is deleted, the `validateDelete` event updates a memo field to indicate that the partner was deleted.

- When adding a new partner or editing an existing partner, the validateInsert and validateLine events ensure that their contribution is set to 100%.

```
/**
 *@NApiVersion 2.x
 *@NScriptType ClientScript
 */
define(['N/error'],
  function(error) {
    function pageInit(context) {
      if (context.mode !== 'create')
        return;
      var currentRecord = context.currentRecord;
      currentRecord.setValue({
        fieldId: 'entity',
        value: 107
      });
    }
    function saveRecord(context) {
      var currentRecord = context.currentRecord;
      if (!currentRecord.getValue({
          fieldId: 'entity'
        }) || currentRecord.getLineCount({
          sublistId: 'item'
        }) < 1)
        throw error.create({
          name: 'MISSING_REQ_ARG',
          message: 'Please enter all the necessary fields on the salesorder before saving'
        });
      return true;
    }
    function validateField(context) {
      var currentRecord = context.currentRecord;
      var sublistName = context.sublistId;
      var sublistFieldName = context.fieldId;
      var line = context.line;
      if (sublistName === 'item') {
        if (sublistFieldName === 'quantity') {
          if (currentRecord.getCurrentSublistValue({
              sublistId: sublistName,
              fieldId: sublistFieldName
            }) < 3)
            currentRecord.setValue({
              fieldId: 'otherrefnum',
              value: 'Quantity is less than 3'
            });
        } else
          currentRecord.setValue({
            fieldId: 'otherrefnum',
            value: 'Quantity accepted'
          });
      }
    }
    return true;
  }
)
```

```

    }
    function fieldChanged(context) {
        var currentRecord = context.currentRecord;
        var sublistName = context.sublistId;
        var sublistFieldName = context.fieldId;
        var line = context.line;
        if (sublistName === 'item' && sublistFieldName === 'item')
            currentRecord.setValue({
                fieldId: 'memo',
                value: 'Item: ' + currentRecord.getCurrentSublistValue({
                    sublistId: 'item',
                    fieldId: 'item'
                }) + ' is selected'
            });
    }
    function postSourcing(context) {
        var currentRecord = context.currentRecord;
        var sublistName = context.sublistId;
        var sublistFieldName = context.fieldId;
        var line = context.line;
        if (sublistName === 'item' && sublistFieldName === 'item')
            if (currentRecord.getCurrentSublistValue({
                sublistId: sublistName,
                fieldId: sublistFieldName
            }) === '39')
                if (currentRecord.getCurrentSublistValue({
                    sublistId: sublistName,
                    fieldId: 'pricelevels'
                }) !== '1-1')
                    currentRecord.setCurrentSublistValue({
                        sublistId: sublistName,
                        fieldId: 'pricelevels',
                        value: '1-1'
                    });
    }
    function lineInit(context) {
        var currentRecord = context.currentRecord;
        var sublistName = context.sublistId;
        if (sublistName === 'partners')
            currentRecord.setCurrentSublistValue({
                sublistId: sublistName,
                fieldId: 'partner',
                value: '55'
            });
    }
    function validateDelete(context) {
        var currentRecord = context.currentRecord;
        var sublistName = context.sublistId;
        if (sublistName === 'partners')
            if (currentRecord.getCurrentSublistValue({
                sublistId: sublistName,
                fieldId: 'partner'
            }) === '55')
                currentRecord.setValue({
                    fieldId: 'memo',

```

```

        value: 'Removing partner sublist'
    });
    return true;
}
function validateInsert(context) {
    var currentRecord = context.currentRecord;
    var sublistName = context.sublistId;
    if (sublistName === 'partners')
        if (currentRecord.getCurrentSublistValue({
            sublistId: sublistName,
            fieldId: 'contribution'
        }) !== '100.0%')
            currentRecord.setCurrentSublistValue({
                sublistId: sublistName,
                fieldId: 'contribution',
                value: '100.0%'
            });
    return true;
}
function validateLine(context) {
    var currentRecord = context.currentRecord;
    var sublistName = context.sublistId;
    if (sublistName === 'partners')
        if (currentRecord.getCurrentSublistValue({
            sublistId: sublistName,
            fieldId: 'contribution'
        }) !== '100.0%')
            currentRecord.setCurrentSublistValue({
                sublistId: sublistName,
                fieldId: 'contribution',
                value: '100.0%'
            });
    return true;
}
function sublistChanged(context) {
    var currentRecord = context.currentRecord;
    var sublistName = context.sublistId;
    var op = context.operation;
    if (sublistName === 'item')
        currentRecord.setValue({
            fieldId: 'memo',
            value: 'Total has changed to ' + currentRecord.getValue({
                fieldId: 'total'
            }) + ' with operation: ' + op
        });
}
return {
    pageInit: pageInit,
    fieldChanged: fieldChanged,
    postSourcing: postSourcing,
    sublistChanged: sublistChanged,
    lineInit: lineInit,
    validateField: validateField,
    validateLine: validateLine,
    validateInsert: validateInsert,
}

```

```

    validateDelete: validateDelete,
    saveRecord: saveRecord
};

});

```

Client Script Entry Points

Script Entry Point	Description
fieldChanged	Executed when a field is changed by a user or client side call.
lineInit	Executed when an existing line is selected.
pageInit	Executed when the page completes loading or when the form is reset.
postSourcing	Executed on transaction forms when a field that sources information from another field is modified.
saveRecord	Executed after the submit button is pressed but before the form is submitted.
sublistChanged	Executed after a sublist has been inserted, removed, or edited.
validateDelete	Executed when removing an existing line from an edit sublist.
validateField	Executes when a field is about to be changed by a user or client side call.
validateInsert	Executed when you insert a line into an edit sublist.
validateLine	Executed before a line is added to an inline editor sublist or editor sublist.

fieldChanged

Description	<p>Executed when a field is changed by a user or client side call. This event may also execute directly through beforeLoad user event scripts. The following sample tasks can be performed:</p> <ul style="list-style-type: none"> ■ Provide the user with additional information based on user input. ■ Disable or enable fields based on user input. <p>For an example, see Client Script Sample.</p> <p>Note: This event does not execute when the field value is changed or entered in the page URL. Use the pageInit function to handle URLs that may contain updated field values. See pageInit.</p>
Returns	void
Since	Version 2015 Release 2

Parameters

Note: The `scriptContext` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
<code>scriptContext.currentRecord</code>	<code>currentRecord.CurrentRecord</code>	The current form record. For more information about <code>CurrentRecord</code> object members, see CurrentRecord Object Members .	Version 2015 Release 2
<code>scriptContext.sublistId</code>	<code>string</code>	The sublist ID name.	Version 2015 Release 2
<code>scriptContext.fieldId</code>	<code>string</code>	The field ID name.	Version 2015 Release 2
<code>scriptContext.line</code>	<code>string</code>	The line number (zero-based index) if the field is in a sublist or a matrix. If the field is not a sublist or matrix, the default value is undefined.	Version 2015 Release 2
<code>scriptContext.column</code>	<code>string</code>	The column number (zero-based index) if the field is in a matrix. If the field is not in a matrix, the default value is undefined.	Version 2015 Release 2

lineInit

Description	Executed when an existing line is selected. This event can behave like a <code>pageInit</code> event for line items in an inline editor sublist or editor sublist. For an example, see Client Script Sample .
Returns	<code>void</code>
Since	Version 2015 Release 2

Parameters

Note: The `scriptContext` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
<code>scriptContext.currentRecord</code>	<code>currentRecord.CurrentRecord</code>	The current form record. For more information about <code>CurrentRecord</code>	Version 2015 Release 2

Parameter	Type	Description	Since
		object members, see CurrentRecord Object Members .	
scriptContext.sublistId	string	The sublist ID name.	Version 2015 Release 2

pageInit

Description	<p>Executed when the page completes loading or when the form is reset. The following sample tasks can be performed:</p> <ul style="list-style-type: none"> ■ Populate field defaults. ■ Disable or enable fields. ■ Change field availability or values depending on the data available for the record. ■ Add flags to set initial values of fields. ■ Provide alerts where the data being loaded is inconsistent or corrupt. ■ Retrieve user login information and change field availability or values accordingly. ■ Validate that fields required for your custom code (but not necessarily required for the form) exist. <p>For an example, see Client Script Sample.</p>
Returns	void
Since	Version 2015 Release 2

Parameters

 **Note:** The scriptContext parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
scriptContext.currentRecord	currentRecord. CurrentRecord	The current form record. For more information about CurrentRecord object members, see CurrentRecord Object Members .	Version 2015 Release 2
scriptContext.mode	string	<p>The mode in which the record is being accessed. The mode can be set to one of the following values:</p> <ul style="list-style-type: none"> ■ copy ■ create ■ edit 	Version 2015 Release 2

 **Note:** For a complete script example, see [Client Script Sample](#).

postSourcing

Description	Executed on transaction forms when a field that sources information from another field is modified. This event behaves like a fieldChanged event after all dependent field values have been set. The event waits for any slaved or cascaded field changes to complete before calling the user defined function. For an example, see Client Script Sample .  Note: The event is not triggered by field changes for a field that does not have any slaved fields.
Returns	void
Since	Version 2015 Release 2

Parameters

 **Note:** The `scriptContext` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
<code>scriptContext.currentRecord</code>	<code>currentRecord</code> <code>CurrentRecord</code>	The current form record. For more information about <code>CurrentRecord</code> object members, see CurrentRecord Object Members .	Version 2015 Release 2
<code>scriptContext sublistId</code>	string	The sublist ID name.	Version 2015 Release 2
<code>scriptContext.fieldId</code>	string	The field ID name.	Version 2015 Release 2

saveRecord

Description	Executed after the submit button is pressed but before the form is submitted. The following sample tasks can be performed: <ul style="list-style-type: none">■ Provide alerts before committing the data.■ Enable fields that were disabled with other functions.■ Redirect the user to a specified URL. For an example, see Client Script Sample .
Returns	Boolean <code>true</code> if the record is valid. Boolean <code>false</code> to suppress form submission.
Since	Version 2015 Release 2

Parameters

Note: The `scriptContext` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
<code>scriptContext.currentRecord</code>	<code>currentRecord</code> . <code>CurrentRecord</code>	The current form record. For more information about <code>CurrentRecord</code> object members, see CurrentRecord Object Members .	Version 2015 Release 2

sublistChanged

Note: The `sublistChanged` function replaces the SuiteScript 1.0 function `recalc`.

Description	Executed after a sublist is inserted, removed, or edited. For an example, see Client Script Sample .
Returns	<code>void</code>
Since	Version 2015 Release 2

Parameters

Note: The `scriptContext` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
<code>scriptContext.currentRecord</code>	<code>currentRecord</code> . <code>CurrentRecord</code>	The current form record. For more information about <code>CurrentRecord</code> object members, see CurrentRecord Object Members .	Version 2015 Release 2
<code>scriptContext.sublistId</code>	<code>string</code>	The sublist ID name.	Version 2015 Release 2

validateDelete

Description	Executed when removing an existing line from an edit sublist. For an example, see Client Script Sample .
Returns	Boolean <code>true</code> if the sublist line is valid. Boolean <code>false</code> to block the removal.
Since	Version 2015 Release 2

Parameters

Note: The `scriptContext` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
<code>scriptContext.currentRecord</code>	<code>currentRecord</code> . <code>CurrentRecord</code>	The current form record. For more information about <code>CurrentRecord</code> object members, see CurrentRecord Object Members .	Version 2015 Release 2
<code>scriptContext.sublistId</code>	<code>string</code>	The sublist ID name.	Version 2015 Release 2

validateField

Description	Executes when a field is about to be changed by a user or client side call. This event executes on fields added in <code>beforeLoad</code> user event scripts. The following sample tasks can be performed: <ul style="list-style-type: none"> ■ Validate field lengths. ■ Restrict field entries to a predefined format. ■ Restrict submitted values to a specified range ■ Validate the submission against entries made in an associated field. For an example, see Client Script Sample .
Returns	Boolean <code>true</code> if the field is valid. Boolean <code>false</code> to prevent the field value from changing.
Since	Version 2015 Release 2

Parameters

Note: The `scriptContext` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
<code>scriptContext.currentRecord</code>	<code>currentRecord</code> . <code>CurrentRecord</code>	The current form record. For more information about <code>CurrentRecord</code> object members, see CurrentRecord Object Members .	Version 2015 Release 2
<code>scriptContext.sublistId</code>	<code>string</code>	The sublist ID name.	Version 2015 Release 2

Parameter	Type	Description	Since
scriptContext.fieldId	string	The field ID name.	Version 2015 Release 2
scriptContext.line	string	The line number (zero-based index) if the field is in a sublist or a matrix. If the field is not a sublist or matrix, the default value is undefined.	Version 2015 Release 2
scriptContext.column	string	The column number (zero-based index) if the field is in a matrix. If the field is not in a matrix, the default value is undefined.	Version 2015 Release 2

validateInsert

Description	Executed when you insert a line into an edit sublist. For an example, see Client Script Sample .
Returns	Boolean <code>true</code> if the sublist line is valid. Boolean <code>false</code> to block the insert.
Since	Version 2015 Release 2

Parameters

Note: The `scriptContext` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
scriptContext.currentRecord	currentRecord. CurrentRecord	The current form record. For more information about CurrentRecord object members, see CurrentRecord Object Members .	Version 2015 Release 2
scriptContext.sublistId	string	The sublist ID name.	Version 2015 Release 2

validateLine

Description	Executed before a line is added to an inline editor sublist or editor sublist. This event can behave like a <code>saveRecord</code> event for line items in an inline editor sublist or editor sublist. For an example, see Client Script Sample .
Returns	Boolean <code>true</code> if the sublist line is valid. Boolean <code>false</code> to reject the operation.

Since	Version 2015 Release 2
-------	------------------------

Parameters

Note: The `scriptContext` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
<code>scriptContext.currentRecord</code>	<code>currentRecord</code> . <code>CurrentRecord</code>	The current form record. For more information about <code>CurrentRecord</code> object members, see CurrentRecord Object Members .	Version 2015 Release 2
<code>scriptContext.sublistId</code>	<code>string</code>	The sublist ID name.	Version 2015 Release 2

Map/Reduce Script Type

- Map/Reduce Script Sample
- Map/Reduce Stages
- Map/Reduce Script Governance
- Check the Status of a Map/Reduce Script
- Map/Reduce Script Troubleshooting
- Map/Reduce Script Type: Best Practices
- Map/Reduce Script Entry Points
- Map/Reduce Script API

Map/reduce scripts provide a structured framework for server-side scripts that process a large number of records or data. When you use this type of script, governance usage and yielding is tracked automatically. If a map/reduce script exceeds a governance limit, script execution are automatically re-scheduled and re-processed. See [API Governance](#) for more details.

The map/reduce script type supports the following deployment options: by schedule, from the **Save and Execute** option on the deployment record, or through the task module (see `task.MapReduceScriptTask`).

In SuiteCloud Plus accounts, map/reduce scripts can process records in parallel. From the script deployment record, you can manually assign the queues to use for distributed processing.

You can monitor the progress of map/reduce scripts by viewing the script status page.

Map/Reduce Script Sample

- [Example 1](#)
- [Example 2](#)

Example 1

The following example shows the basic framework for a map/reduce script. This script provides a word count for text.

The input stage fetches a line of text. In the map stage, logic determines the parts of text that are not words to ensure that punctuation is not included in the count. The reduce stage is used to write the text and corresponding word count. In the summary stage, the word count for the input text is saved as a file.

```
/**
 * @NApiVersion 2.x
 * @NScriptType MapReduceScript
 */
define(['N/file'],
    function(file) {
        const PUNCTUATION_REGEX = /[ \u2000-\u206F\u2E00-\u2E7F\!\"#\$\%&\(\)\*\+\,-\.\/;:<=>\?\@\[\]\^_`{\}\|\~]/g;
        function getInputData() {
            return "the quick brown fox \njumped over the lazy dog.".split('\n');
        }
        function map(context) {
            for (var i = 0; context.value && i < context.value.length; i++) {
                if (context.value[i] !== ' ' && !PUNCTUATION_REGEX.test(context.value[i]))
                    context.write(context.value[i], 1);
            }
        }
        function reduce(context) {
            context.write(context.key, context.values.length);
        }
        function summarize(summary) {
            var type = summary.toString();
            log.audit(type + ' Usage Consumed', summary.usage);
            log.audit(type + ' Concurrency Number ', summary.concurrency);
            log.audit(type + ' Number of Yields', summary.yields);
            var contents = '';
            summary.output.iterator().each(function(key, value) {
                contents += (key + ' ' + value + '\n');
                return true;
            });
            var fileObj = file.create({
                name: 'wordCountResult.txt',
                fileType: file.Type.PLAINTEXT,
                contents: contents
            });
            fileObj.folder = -15;
            fileObj.save();
        }
        return {
            getInputData: getInputData,
            map: map,
            reduce: reduce,
            summarize: summarize
        };
    });
});
```

Example 2

The following example shows a sample script that processes invoices and contains logic to handle errors. This script is designed to do the following:

- Find the ids and associated customers for all open invoices.
- Apply a location-based discount to each invoice.
- Write each invoice to the reduce stage so it is grouped by customer id.
- Initialize a new CustomerPayment for each customer applied only to the invoices specified in the reduce values.
- Create a custom record capturing the details of which records were processed.
- Notify administrators of any exceptions via an email notification.

Prior to running this sample, you need to manually create a custom record type with id "customrecord_summary", and text fields with id "custrecord_time", "custrecord_usage", and "custrecord_yields".

Script Sample Prerequisites

1. From the NetSuite UI, select Customization > List, Records, & Fields > Record Types > New.
2. From the Custom Record Type page, enter a value for name.
3. In the ID field, enter "customrecord_summary".
4. Select Save.
5. From the Fields subtab, do the following:
 - Select New Field. Enter a label and set ID to "custrecord_time". Ensure that the Type field is set to Free-Form Text. Select Save & New.
 - Select New Field. Enter a label and set ID to "custrecord_usage". Ensure that the Type field is set to Free-Form Text. Select Save & New.
 - Select New Field. Enter a label and set ID to "custrecord_yields". Ensure that the Type field is set to Free-Form Text. Select Save.

```
/** 
 * @NApiVersion 2.0
 * @NScriptType MapReduceScript
 */
define(['N/search', 'N/record', 'N/email', 'N/runtime', 'N/error'],
    function(search, record, email, runtime, error)
{
    function handleErrorAndSendNotification(e, stage)
    {
        log.error('Stage: ' + stage + ' failed', e);

        var author = -5;
        var recipients = 'notify@company.com';
        var subject = 'Map/Reduce script ' + runtime.getCurrentScript().id + ' failed for s
tage: ' + stage;
        var body = 'An error occurred with the following information:\n' +
            'Error code: ' + e.name + '\n' +
            'Error msg: ' + e.message;

        email.send({
            author: author,
            recipients: recipients,
            subject: subject,
            body: body
        });
    }
})
```

```

function handleErrorIfAny(summary)
{
    var inputSummary = summary.inputSummary;
    var mapSummary = summary.mapSummary;
    var reduceSummary = summary.reduceSummary;

    if (inputSummary.error)
    {
        var e = error.create({
            name: 'INPUT_STAGE_FAILED',
            message: inputSummary.error
        });
        handleErrorAndSendNotification(e, 'getInputData');
    }

    handleErrorInStage('map', mapSummary);
    handleErrorInStage('reduce', reduceSummary);
}

function handleErrorInStage(stage, summary)
{
    var errorMsg = [];
    summary.errors.iterator().each(function(key, value){
        var msg = 'Failure to accept payment from customer id: ' + key + '. Error was:
' + JSON.parse(value).message + '\n';
        errorMsg.push(msg);
        return true;
    });
    if (errorMsg.length > 0)
    {
        var e = error.create({
            name: 'RECORD_TRANSFORM_FAILED',
            message: JSON.stringify(errorMsg)
        });
        handleErrorAndSendNotification(e, stage);
    }
}

function createSummaryRecord(summary)
{
    try
    {
        var seconds = summary.seconds;
        var usage = summary.usage;
        var yields = summary.yields;

        var rec = record.create({
            type: 'customrecord_summary',
        });

        rec.setValue({
            fieldId : 'name',
            value: 'Summary for M/R script: ' + runtime.getCurrentScript().id
        });
    }
}

```

```

        rec.setValue({
            fieldId: 'custrecord_time',
            value: seconds
        });
        rec.setValue({
            fieldId: 'custrecord_usage',
            value: usage
        });
        rec.setValue({
            fieldId: 'custrecord_yields',
            value: yields
        });

        rec.save();
    }
    catch(e)
    {
        handleErrorAndSendNotification(e, 'summarize');
    }
}

function applyLocationDiscountToInvoice(recordId)
{
    var invoice = record.load({
        type: record.Type.INVOICE,
        id: recordId,
        isDynamic: true
    });

    var location = invoice.getText({
        fieldId: 'location'
    });

    var discount;
    if (location === 'East Coast')
        discount = 'Eight Percent';
    else if (location === 'West Coast')
        discount = 'Five Percent';
    else if (location === 'United Kingdom')
        discount = 'Nine Percent';
    else
        discount = '';

    invoice.setText({
        fieldId: 'discountitem',
        text: discount,
        ignoreFieldChange : false
    });
    log.debug(recordId + ' has been updated with location-based discount.');
    invoice.save();
}

function getInputData()
{
    return search.create({

```

```

        type: record.Type.INVOICE,
        filters: [['status', search.Operator.IS, 'open']],
        columns: ['entity'],
        title: 'Open Invoice Search'
    });
}

function map(context)
{
    var searchResult = JSON.parse(context.value);
    var invoiceId = searchResult.id;
    var entityId = searchResult.values.entity.value;

    applyLocationDiscountToInvoice(invoiceId);

    context.write(entityId, invoiceId);
}

function reduce(context)
{
    var customerId = context.key;

    var custPayment = record.transform({
        fromType: record.Type.CUSTOMER,
        fromId: customerId,
        toType: record.Type.CUSTOMER_PAYMENT,
        isDynamic: true
    });

    var lineCount = custPayment.getLineCount('apply');
    for (var j = 0; j < lineCount; j++)
    {
        custPayment.selectLine({
            sublistId: 'apply',
            line: j
        });
        custPayment.setCurrentSublistValue({
            sublistId: 'apply',
            fieldId: 'apply',
            value: true
        });
    }

    var custPaymentId = custPayment.save();
    context.write(custPaymentId);
}

function summarize(summary)
{
    handleErrorIfAny(summary);
    createSummaryRecord(summary);
}

return {
    getInputData: getInputData,
}

```

```

        map: map,
        reduce: reduce,
        summarize: summarize
    );
});

```

Map/Reduce Stages

The map/reduce script type goes through at least two of five possible stages.

The stages are processed in the following order. Note that each stage must complete before the next stage begins.

- Get Input Data – Acquires a collection of data. This stage is always processed first and is required. The input stage runs sequentially.
- Map – Parses each row of data into a key/value pair. One pair (key/value) is passed per function invocation. If this stage is skipped, the reduce stage is required. Data is processed in parallel in this stage.
- Shuffle – Groups values based on keys. This is an automatic process that always follows completion of the map stage. There is no direct access to this stage as it is handled by the map/reduce script framework. Data is processed sequentially in this stage.
- Reduce – Evaluates the data in each group. One group (key/values) is passed per function invocation. If this stage is skipped, the map stage is required. Data is processed in parallel in this stage.
- Summarize – Summarizes the output of the previous stages. Developers can use this stage to summarize the data from the entire map/reduce process and write it to a file or send an email. This stage is optional and is not technically a part of the map/reduce process. The summarize stage runs sequentially.



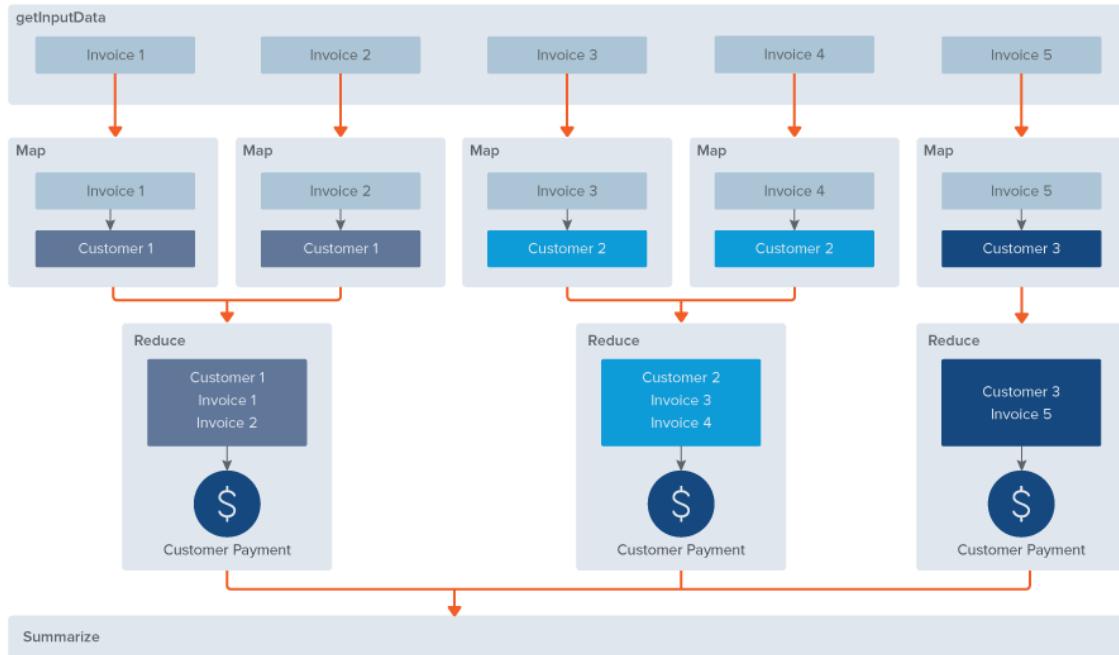
Note: It is not mandatory to use both the map stage and the reduce stage. You may skip one of those stages.

The following diagram illustrates these stages, in the context of processing a set of invoices.

In this example, the stages are used as follows:

- Get Input Data – A collection of invoices that require payment is loaded.
- Map – Each invoice is paired to the customer expected to pay it. The key/value pairs are returned, where customerID is the key and the invoice is the value. For five invoices, the map is invoked five times.
- Reduce – There are three unique groups of invoices based on customerID. For three groups, reduce is invoked three times. To create a customer payment for every group, custom logic iterates over each group using customerID as the key.
- Summarize – Custom logic fetches various metrics (for example, number of invoices paid) and sends the output as an email notification.

For a code sample similar to this example, see [Example 2](#).



Passing Data to a Map or Reduce Stage

To prevent unintended alteration of data when it is passed between stages, key/value pairs are always serialized into strings. For map/reduce scripts, SuiteScript 2.0 checks if the data passed to the next stage is a string, and uses `JSON.stringify()` to convert the key or value into a string as necessary.

Objects serialized to JSON remain in JSON format. To avoid possible errors, SuiteScript does not automatically deserialize the data. For example, an error might result from an attempt to convert structured data types (such as CSV or XML) that are not valid JSON. At your discretion, you can use `JSON.parse()` to convert the JSON string back into a native JS object.

Map/Reduce Script Governance

API Governance

The following table lists the applicable governance limits for map/reduce scripts.

When the script reaches a usage limit for a map/reduce task instance, the queue is automatically yielded.

Stage	API Usage Limit
Input	<ul style="list-style-type: none"> ■ 10,000 units
Map	<ul style="list-style-type: none"> ■ 1,000 units per function invocation ■ 10,000 units per instance of a map/reduce task
Reduce	<ul style="list-style-type: none"> ■ 5,000 units per function invocation ■ 10,000 units per instance of a map/reduce task
Summarize	<ul style="list-style-type: none"> ■ 10,000 units



Important: Yielding is also subject to the Yield After Min setting on the script deployment record. See [Change the Time to Yield](#).

Script Instance Governance

The total persisted size of data for a map/reduce script is not allowed to exceed 50MB. This includes the amount of search results stored.

In a search result, each key and the serialized size of each value is counted towards the total size. Note that value size is proportional to the number of search result columns in a result set.

During the map or reduce stage, the total size is a measure of the amount of keys and values to be processed. After a key or value is processed, it is not counted towards the total size.

If the limit is exceeded, that instance of a map/reduce task fails and a STORAGE_SIZE_EXCEEDED error is thrown.

Yielding



Important: It is not possible to yield a *running* script in SuiteScript 2.0. Map/reduce yields can occur only in the map or reduce stages, after a map() or reduce() function is fully executed.

The usage and persisted size limits are applied per function invocation. For the getInput and summarize stages, there is a direct correspondence of function-to-stage because these stages execute the function only once.

However, the map and reduce stages can execute many map() and reduce() functions. Even a single map or reduce job can execute many functions. It's possible for a map/reduce script to run a long time if it doesn't violate the 50MB persisted size limit. For example, a reduce job can easily consume more than 5000 usage units and not fail, because the limit is per each executed reduce() function.

Therefore, avoid putting a massive amount of logic in a map and reduce function. The map and reduce function should do a unit of work to be applied many times over each data piece.

Check the Status of a Map/Reduce Script

- [View Map/Reduce Script Status](#)
- [View Details of Script Instances](#)
- [Programmatically Retrieve Script Instance Details](#)

View Map/Reduce Script Status

You can monitor map/reduce script execution via the map/reduce script status page in the UI. With the script status page, you can see whether a map/reduce script deployment is pending, in progress, or unable to complete.

If all tasks are pending, you can cancel a script deployment from this page.

From NetSuite, go to Customization > Scripting > Map/Reduce Script Status.

To help you understand and optimize the performance of script entry points used, you can drill down for more details about map stages, processing utilization, and timing. You can use this information to gain insight about the time required to complete a stage or process a task.

If you are a SuiteCloud Plus user, reviewing these details can help you to isolate a problem to a particular queue. Use the links on the script status page to jump back to the deployment record and edit queue assignments.

View Details of Script Instances

A script instance is the individual job associated with a submitted task.

From the Map/Reduce Script Status page, click View Details.

Each row contains information about an individual map/reduce task per stage and per queue.

Programmatically Retrieve Script Instance Details

To get the script status via the [N/search Module](#), create and load a search using `scheduledscriptinstance` as the type argument.

To get the status via the [N/task Module](#), see `task.MapReduceScriptTask`.

Change the Time to Yield

To prevent a particular script from monopolizing your processing resources, you can set a time limit to reduce the amount of time before a yield occurs on a per-script basis.

This is available as a UI setting on the deployment record so that you can enforce a time limit on a script you do not own.

1. Go to Customization > Scripting > Scripts > Script Deployments.
2. If necessary, change the filters to show the script you need.
3. Click Edit.
4. In the Yield After Mins field, enter an amount of time equalling 60 minutes or less.

Map/Reduce Script Troubleshooting

You have several options to learn more about how your Map/Reduce script is running:

- Make a separate test suite as you develop and modify your Map/Reduce script. See [Ad-hoc Debugging](#)
- Check the execution logs. See [N/log Module](#).
- Monitor the script status. See [Check the Status of a Map/Reduce Script](#) and [Programmatically Retrieve Script Instance Details](#).
- Detect any server restarts that interrupted map/reduce script execution. See [GetInputContext.isRestarted](#), [MapContext.isRestarted](#), [ReduceContext.isRestarted](#), and [Summary.isRestarted](#).

Do not use the SuiteScript Debugger for deployed debugging of a Map/Reduce script type. However, you may wish to test any dependencies on other types of scripts. Remember that to test existing scripts, the Script Deployment Status must be set to Testing and the currently logged in user must be listed as the script record owner. For information about the SuiteScript Debugger, see the help topic [SuiteScript Debugger](#).

Ad-hoc Debugging

 **Note:** You cannot use the SuiteScript Debugger for server-side script debugging of a Map/Reduce script.

To ad-hoc test a Map/Reduce script, NetSuite recommends splitting the script into entry point level sections. Use the sections to form unit tests. Each section should function as an entry point script that

can be executed without external dependencies. If passing in modules, make sure that you use the require function and absolute paths.

To test map or reduce stages, you will need to create a mock context that seeds values and provides the dependent objects and parameters. Then, check the states, behavior, inputs, and outputs of map/reduce functions using assertion statements. Note that the getInputData stage does not take parameters, so it will not require mock context and can be tested more conventionally.

To test a summarize stage, if it contains logic operating on a final set of data that has no return, use assertions and logs to gather information.

To assist development of your unit test, download the SuiteScript 2.0 API files, specifically those representing the [mapReduce.MapContext](#), [mapReduce.ReduceContext](#), and [mapReduce.Summary](#) objects. These file can act as a schema for the properties and methods you want to test or mock. To access the files, do the following:

1. From NetSuite, select Documents > File Cabinet.
2. Select SuiteScript 2.0 API to download the zip folder.
3. Extract the mapReduceContext and mapReduceSummary .js files.

Map/Reduce Script Type: Best Practices

- Passing Search Data to the getInput Stage
- Optimizing Processing Speed during Map and Reduce Stages
- Allowing Other Scripts to Run During Map/Reduce Script Execution
- Handling Server Restarts in Map/Reduce Scripts

Passing Search Data to the getInput Stage

Instead of running a search to gather large amounts of data, consider using an object reference or saved search to pass data into the getInput stage. This enables you to utilize the map/reduce framework to generate the input data, and extends the timeout limit applied to search execution. The map/reduce framework can run the search for you and has access to a longer search timeout.

For a code snippet that shows passing in a saved search, see the [Syntax](#) sample for `getInputData()`. Note that if you already have a saved search, you can return a reference to a saved search without needing to load it.

For more information about passing in an object reference, see [getInputData\(\)](#).

Optimizing Processing Speed during Map and Reduce Stages

SuiteScript 2.0 includes a buffer for the map/reduce script type which you can use to enhance performance. Database storage of map and reduce outputs can result in significant overhead. The buffer enables you to specify the number of outputs stored simultaneously. Higher buffer values maximize performance. Lower buffer values reduce risks to data in case of a server restart.

When the script executes, the buffer is applied to the map and reduce entry point functions. The buffer value determines when the data is stored. For example, a buffer size of 4 would result in data being stored after every fourth map or reduce function.

From the map/reduce script deployment record, you can set the buffer to one of the following values: 1, 2, 4, 8, 16, 32, or 64.

The screenshot shows the 'Script Deployment' page. On the left, there are fields for 'SCRIPT' (MR Sample), 'TITLE' (MR Sample), and 'ID' (customdeploy1). A checkbox labeled 'DEPLOYED' is checked. On the right, there are sections for 'STATUS' (Not Scheduled), 'LOG LEVEL' (Debug), 'EXECUTE AS ROLE' (Administrator), and 'YIELD AFTER MINUTES' (60). Below these is a dropdown menu titled 'BUFFER SIZE' with values 1, 2, 4, 8, 16, 32, and 64. The value '1' is selected and highlighted with a red border.



Important: When you set the buffer size, you need to strike a balance between performance and the risk of data loss. The higher you set the buffer size, the lower the overhead, but the greater the risk of data loss. Higher buffer values are recommended for fast, algorithmic operations. Higher buffer values are not recommended for record handling operations. In case of application server crash, functions without output data stored have to be repeated, and the repeat of record handling operations may result in duplicate data being saved. To avoid this situation, make sure to use a lower buffer value for record handling operations. You can detect a restart that impacted a script using the `isRestarted` property for entry point functions.

Allowing Other Scripts to Run During Map/Reduce Script Execution

Map/Reduce scripts are run in stages. You can run all stages continuously or leave gaps that allow you to run additional scripts. By default, the stages can run continuously (unless yielding occurs).



Note: The setting does not impact script yielding. If the script yields during the map or reduce stages, other scripts can run where it yielded.

To allow gaps between stages, from the Script Deployment record, clear the Queue All Stages At Once check box.



Note: If the account uses SuiteCloud Plus and the script is deployed for parallel execution, consider allowing gaps. This prevents idle processing power during the `getInput` and `shuffle` stages.

Handling Server Restarts in Map/Reduce Scripts

Occasionally, a script failure may occur due to an application server restart. This could be due to a NetSuite update, NetSuite maintenance, or an unexpected failure of the execution environment.

Restarts can terminate an application forcefully at any moment. Therefore, robust scripts must account for restarts and be able to recover from an unexpected interruption.

In a map/reduce script, each restarted piece of the script will automatically delete any internal map/reduce data that this piece created (for example, the key/value pairs that drive the execution of a entire mapping task). However, you must develop your own code to handle any parts of the script that modify additional data (for example, creation of NetSuite records like sales orders), which is *never* automatically deleted.

See the following topics to learn more about how restarts and map/reduce script execution:

- [Example 3: Design of a Robust Map/Reduce Script](#)
- [Example 4: A Problematic Map/Reduce Script](#)
- [Example 5: A Robust Map/Reduce Script](#)
- [Execution of Restarted Map/Reduce Stages](#)

Example 3: Design of a Robust Map/Reduce Script

The following script is designed to detect restarts at particular stages in processing, and to hold logic to run in the event of a restart.

Consider this example as a basic template, where the comment `// I might do something differently` denotes implementation of a special function for each stage, to ensure that the script can repeat itself with the same result. Or, to run a recovery task, such as removing duplicate records.

The script includes a check on the `isRestarted` property for each entry point function. If the value of `isRestarted` is `true`, the example script shows a placeholder for invoking a function. This is a meant as a placeholder where implementation of logic for protection against restarts and data errors could be inserted.

For more information about an interrupted map/reduce stage, see [Execution of Restarted Map/Reduce Stages](#).

```
define([], function(){
    return {
        getInputData: function (context)
        {
            if (context.isRestarted)
            {
                // I might do something differently
            }
            .
            .
            .
            return inputForNextStage;
        },
        map: function (context)
        {
            if (context.isRestarted)
            {
                // I might do something differently
            }
            .
            .
            .
        }
    }
});
```

```

},
reduce: function (context)
{
    if (context.isRestarted)
    {
        // I might do something differently
    }
    .
    .
    .
},
summarize: function (summary)
{
    if (summary.isRestarted)
    {
        // I might do something differently
    }
    .
    .
    .
}
});
);
});
```

Example 4: A Problematic Map/Reduce Script

The purpose of this script is to perform a search and process the results. However, it is not adequately prepared for an unexpected restart. The script still needs logic to help prevent an unrecoverable state and prevent creation of erroneous or duplicate data during re-execution.

In Example 4, if the script is forcefully interrupted during the map stage, some sales orders might be updated twice when the map function is re-executed. See [Example 5: A Robust Map/Reduce Script](#) for an improved example.

Note that the other stages in this script do not require improvement for handling a restart. If the get input stage is re-executed, the map/reduce framework ensures that each result of the search is passed to the map stage only one time. In this script, the getInputStream stage does not change any additional data, so no special restart logic is needed to ensure correct updates of getInputStream data. Likewise, the reduce and summarize stages do not change any additional data. They process only internal map/reduce data.

```

/**
 * @NApiVersion 2.0
 * @NScriptType mapreducescript
 */
define(['N/search', 'N/record'],
function(search, record){
    return {
        getInputData: function (context)
        {
            var filter1 = search.createFilter({
                name: 'mainline',
                operator: search.Operator.IS,
                values: true
            });
        }
    };
});
```

```

        var column1 = search.createColumn({name: 'recordtype'});
        var srch = search.create({
            type: search.Type.SALES_ORDER,
            filters: [filter1],
            columns: [column1]
        });
        return srch;
    },
    map: function (context)
    {
        var soEntry = JSON.parse(context.value);
        var so = record.load({
            type: soEntry.values.recordtype,
            id: context.key
        });
        // UPDATE so FIELDS
        so.save();

        context.write(soEntry.values.recordtype, context.key);
    },
    reduce: function (context)
    {
        context.write(context.key, context.values.length);
    },
    summarize: function (summary)
    {
        var totalRecordsUpdated = 0;
        summary.output.iterator().each(function (key, value)
        {
            log.audit({
                title: key + ' records updated',
                details: value
            );
            totalRecordsUpdated += parseInt(value);
            return true;
        });
        log.audit({
            title: 'Total records updated',
            details: totalRecordsUpdated
        });
    }
}
});
```

Example 5: A Robust Map/Reduce Script

Comparing Example 5 to Example 4, a filter was added to the search in the `getInput` stage. The purpose is to filter out already processed sales orders. The filter makes it possible to re-execute the whole map/reduce task repeatedly, because when the whole task is re-executed, the additional filter ensures that only unprocessed sales orders will be returned from the input stage and not all sales orders.

There are also substantial improvements to the map function. In Example 5, if the `((context.isRestarted === false))` condition is met, the script knows it is the first execution of the map function for the current key/value pair. It won't need to perform any additional checks and can go directly to the sales order record update.

During the sales order update, an operation sets the `custbody_processed_flag` flag. The script performs a check on this flag only as necessary. If (`context.isRestarted === true`), then the script looks up the appropriate processed flag value, and executes the sales order update only if it wasn't already updated.

Although the script includes more checks and lookups than example 4, the processing demand is light. To perform the check, the script uses a lookup method that doesn't load the full record. If the processed flag value is `true`, then the record is not loaded again.

In the map function, note that the `context.write(...)` statement is not in the if-statement body. It is because when a map function for a particular key/value pair is restarted, all these writes done in the previous execution of the map function are deleted. So there is no need to check which writes have or haven't been done.

The reduce function is not changed from Example 4. This reduce stage handles only the map/reduce internal data, and so the map/reduce framework ensures that even when the reduce function is restarted for a particular key/value pair, only the writes from its last execution for the key/value pair are passed to the next stage.

The summarize function also didn't require improvement. However, it is a good practice to log any restarts. For example, to account for when the "Total records updated" entry appears twice in the execution log for a single map/reduce task execution.

```
/*
 * @NApiVersion 2.0
 * @NScriptType mapreducescript
 */
define(['N/search', 'N/record'],
    function(search, record){
    function(search, record){
        return {
            getInputData: function (context)
            {
                var filter1 = search.createFilter({
                    name: 'mainline',
                    operator: search.Operator.IS,
                    values: true
                });
                var filter2 = search.createFilter({
                    name: 'custbody_processed_flag',
                    operator: search.Operator.IS,
                    values: false
                });
                var column1 = search.createColumn({name: 'recordtype'});
                var srch = search.create({
                    type: search.Type.SALES_ORDER,
                    filters: [filter1, filter2],
                    columns: [column1]
                });
                return srch;
            },
            map: function (context)
            {
                var soEntry = JSON.parse(context.value);
                var alreadyProcessed = false;
                if (context.isRestarted)
                {
                    var lookupResult = search.lookupFields({

```

```

        type: soEntry.values.recordtype,
        id: context.key,
        columns: ['custbody_processed_flag']
    });
    alreadyProcessed = lookupResult.custbody_processed_flag;
}
if (!alreadyProcessed)
{
    var so = record.load({
        type: soEntry.values.recordtype,
        id: context.key
    });
    // UPDATE so FIELDS
    so.setValue({
        fieldId: 'custbody_processed_flag',
        value: true
    });
    so.save();
}

context.write(soEntry.values.recordtype, context.key);
},
reduce: function (context)
{
    context.write(context.key, context.values.length);
},
summarize: function (summary)
{
    if (summary.isRestarted)
    {
        log.audit({details: 'Summary stage is being restarted!'});
    }
    var totalRecordsUpdated = 0;
    summary.output.iterator().each(function (key, value)
    {
        log.audit({
            title: key + ' records updated',
            details: value
        });
        totalRecordsUpdated += parseInt(value);
        return true;
    });
    log.audit({
        title: 'Total records updated',
        details: totalRecordsUpdated
    });
}
}
);

```

Execution of Restarted Map/Reduce Stages

Keep in mind that processing a map/reduce script can involve many jobs. The getInputs, shuffle, and summarize stages are each processed with a single job. However, any number of jobs can participate

in the map and reduce stages. Within a map stage or a reduce stage, jobs can run in parallel. Any of the jobs can be forcefully terminated at any moment. The impact of this event depends on the status of the job (what it was doing), and in which stage it was running.

For details, see the following topics:

- [Termination of getInput Stage](#)
- [Termination of Shuffle Stage](#)
- [Termination of Parallel Stages](#)
- [Termination of Summarize Stage](#)

Termination of getInput Stage

The work of a serial stage (getInput, shuffle, and summarize stages) is done in a single job. If the getInput stage job is forcefully terminated, it is later restarted. The getInput portion of the script can find out whether it is the restarted execution by examining the `isRestarted` attribute of the context argument (`GetInputContext.isRestarted`). The script is being restarted if and only if `(context.isRestarted === true)`.

Note that the input for the next stage is computed from the return value of the getInput script. Next stage input is written after the getInput stage finishes. Therefore, even the restarted getInput script is expected to return the same data. The map/reduce framework helps to ensure that no data is written twice.

However, if the getInput script is changing some additional data (for example, creating NetSuite records), it should contain code to handle duplicated processing. The script needs idempotent operations to ensure that these records are not created twice, if this is undesired.

Termination of Shuffle Stage

The shuffle stage does not contain any custom code, so if the shuffle stage job is forcefully terminated, it is later restarted and all the work is completely redone. There is no impact other than that the stage takes longer to finish.

Termination of Parallel Stages

Map and reduce stages can execute jobs in parallel, so they are considered parallel stages. An application restart will affect parallel stages in the same way. The following example covers impact of restart during the map stage. Note that termination of a reduce stage will behave very similarly.

The purpose of a map stage is to execute a map function on each key/value pair supplied by the previous stage (getInput). Multiple jobs participate in the map stage. Map jobs will claim key/value pairs (or a specific number of key/value pairs) for which the map function was not executed yet. The job sets a flag for these key/value pairs so that no other job can execute the map function on them. Then, the job sequentially executes the map function on the key/value pairs it flagged. The map stage is finished when the map function is executed on all key/value pairs.

The number of jobs that can participate on the map stage is unlimited. Only the maximum concurrency is limited. Initially, the number of map jobs is equal to the selected concurrency in the corresponding map/reduce script deployment. However, to prevent a single map/reduce task from monopolizing all computational resources in the account, each map job can yield itself to allow other jobs to execute. The yield creates an additional map job and the number of yields is unlimited.

Note: This is a different type of yield compared to yield in a SuiteScript 1.0 scheduled script. In SuiteScript 1.0, the yield happens in the middle of a script execution. In a map job, the yield can happen only between two map function executions, and not in the middle of one.

If a map job is forcefully terminated, it is later restarted. First, the job executes the map function on all key/value pairs that it took and did not mark finished before termination. It is the only map job that

can execute the map function on those pairs. They cannot be taken by other map job. After those key/value pairs are processed, the map job continues normally (takes other unfinished key/value pairs and executes the map function on them).

In some cases, the map function can be re-executed on multiple key/value pairs. The number of pairs that a map function can re-execute will depend on the buffer size selected on the deployment page. The buffer size determines the number of key/value pairs originally taken in a batch. The job marks the batch as finished only when the map function is executed on all of them. Therefore, if the map job is forcefully terminated in the middle of the batch, the entire batch will be processed from the beginning when the map job is restarted.

Note that the map/reduce framework deletes all key/value pairs written from a partially-executed batch, so that they are not written out twice. Therefore, the map function does not need to check whether or not `MapContext.write(key,value)` for a particular key/value has already been executed. However, if the map function is changing some additional data, it must also be designed to use idempotent operations. For example, if a map function created NetSuite records, the script should perform additional checks to ensure that these records are not created twice, if this is undesired.

To check if a map function execution is a part of a restarted batch, the script must examine the `isRestarted` attribute in the context argument (`MapContext.isRestarted`). The map function is in the restarted batch if and only if `(context.isRestarted === true)`.

Be aware that a restarted value of `true` is only an indication that some part of the script might have already been executed. Even if `context.isRestarted === true`, a map function could run on a particular key/value for the first time. For example, the map job was forcefully terminated after the map job took the key/value pair for processing, but before it executed the map function on it. This is more likely to occur if a high buffer value is set on the map/reduce deployment. For more information about the buffer, see [Optimizing Processing Speed during Map and Reduce Stages](#).

Termination of Summarize Stage

If the summarize stage job is forcefully terminated, it is later restarted. The summarize portion of the script can find out whether it is the restarted execution by examining the `isRestarted` attribute of the summary argument (`Summary.isRestarted`).

The script is being restarted if and only if `(summary.isRestarted === true)`.

Map/Reduce Script Entry Points

Map/Reduce Script Entry Points

The map/reduce script type includes four entry points that control the script's flow into each map/reduce stage. See [Map/Reduce Stages](#).

You must provide the initial data collection for the first stage. For the remaining script entry points, you do not directly provide input data (contained in the `MapContext`, `ReduceContext`, or `Summary` Objects). This is handled by the framework based on the original input data passed to `getInputData()`.

<code>getInputData()</code>	Marks the beginning of the map/reduce process. Invokes the input stage. This entry point is required. Also see Passing Search Data to the getInput Stage .
<code>map(mapContext)</code>	Invokes the map stage.

	This entry point is optional. If this entry point is skipped, <code>reduce</code> is required.
<code>reduce(reduceContext)</code>	Invokes the reduce stage. This entry point is optional. If this entry point is skipped, <code>map</code> is required.
<code>summarize(summary)</code>	Invokes the summarize stage. This entry point is optional.

getInputData()

Description	<p>Marks the beginning of the map/reduce process. The purpose of the input stage is to generate the input data.</p> <p>Executes when the <code>getInputData</code> entry point is triggered. This entry point is required.</p> <p>Note: When returning a <code>mapReduce.ObjectRef</code>, the supported types are either <code>search</code> or <code>file</code>.</p> <p>Note: When <code>getInputData ()</code> returns a data structure with a non-string value, before the value is stored, it is converted to a JSON string with <code>JSON.stringify()</code>.</p> <p>Also see Passing Search Data to the getInput Stage.</p>
Returns	<p>Array Object <code>search.Search</code> <code>mapReduce.ObjectRef</code> <code>file.File</code> Object</p> <p>Examples:</p> <ul style="list-style-type: none"> ■ Array <pre>[{a : 'b'},{c : 'd'}]</pre> ■ Object <pre>{ a: {...}, b: {...} }</pre> ■ <code>search.Search</code> Object <pre>search.load({ id: 1234 })</pre> ■ Object Reference <pre>{ type: 'search', id: 1234 }</pre> ■ A <code>file.File</code> Object

	<pre>file.load({ id: 1234 })</pre> <p>■ Object Reference</p> <pre>{ type: 'file', path: '/SuiteScripts/data/names.txt' }</pre>
Since	Version 2015 Release 2

Errors

When an error is thrown in this function, the job proceeds to the [summarize\(summary\)](#) function. The serialized error is encapsulated in the [InputSummary.error](#) property.

Syntax

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

```
...
getInputData: function ()
{
    return {
        type:'search',
        id:1234
    }; //Saved Search ID - returns all Sales Orders in Pending Fulfillment state
},
...
```

map(mapContext)

Description	<p>Executes when the <code>map</code> entry point is triggered. When you add custom logic to this entry point function, that logic is applied to each key/value pair. One key/value pair is processed per function invocation. When a search is given as input data, one call is made per row. When a file is given as input data, one call is made per line.</p> <p>Note: The output of the map is a set of key/value pairs. During the shuffle stage that always follows, these pairs are automatically grouped by key.</p>
Returns	Void
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description
mapReduce.MapContext	Object	Required	Data collection containing the key/value pairs to process through the map stage.

Errors

When an error is thrown, the map/reduce task proceeds to the next key/value pair in a `mapReduce.MapContext` Object. The serialized error can be accessed using `Iterator.iterator().each(iteratorFunction)` during the summarize stage.

Syntax

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

```
...
function map(context)
{
    for (var i = 0; context.value && i < context.value.length; i++)
        if (context.value[i] !== ' ' && !PUNCTUATION_REGEX.test(context.value[i]))
            context.write(context.value[i], 1);
}

...
```

reduce(reduceContext)

Description	Executes when the <code>reduce</code> entry point is triggered. When you add custom logic to this entry point function, that logic is applied to each group. One group (key/values) is passed per function invocation. Note: If you are using values from the map stage, they have been automatically grouped by key. The grouping occurs during the shuffle stage, before reduce calls are invoked.
Returns	Void
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description
mapReduce.ReduceContext	Object	Required	Data collection containing the groups to process through the reduce stage.

Errors

When an error is thrown, the map/reduce task proceeds to the next key/value pair in a `mapReduce.ReduceContext` Object. The serialized error can be accessed using `Iterator.iterator().each(iteratorFunction)` during the summarize stage.

Syntax

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

```
...
function reduce(context)
{
    context.write(context.key, context.values.length);
}
...
```

summarize(summary)

Description	Executes when the <code>summarize</code> entry point is triggered. When you add custom logic to this entry point function, that logic is applied to the result set.
Returns	<code>Void</code>
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description
<code>summary</code>	<code>mapReduce.Summary</code>	Required	Holds statistics regarding the execution of a map/reduce script.

Syntax

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

```
...
function summarize(summary)
{
    var type = summary.toString();
    log.audit(type + ' Usage Consumed', summary.usage);
    log.audit(type + ' Concurrency Number ', summary.concurrency);
    log.audit(type + ' Number of Yields', summary.yields);
    var contents = '';
    summary.output.iterator().each(function (key, value)
```

```

    {
        contents += (key + ' ' + value + '\n');
        return true;
    });
...

```

Map/Reduce Script API

- Map/Reduce Object Members
- InputSummary Object Members
- Iterator Object Members
- MapContext Object Members
- MapSummary Object Members
- ObjectRef Object Members
- ReduceContext Object Members
- ReduceSummary Object Members
- Summary Object Members

Map/Reduce Object Members

Member Type	Name	Description
Object	mapReduce.GetInputContext	Holds the <code>GetInputContext.isRestarted</code> property
	mapReduce.InputSummary	Holds statistics regarding the input stage
	mapReduce.Iterator	Provides the keys and values written as output during the reduce stage.
	mapReduce.MapContext	Contains a single key/value pair to process through the map stage. Passed in when <code>map(mapContext)</code> is called
	mapReduce.MapSummary	Holds statistics regarding the map stage.
	mapReduce.ObjectRef	Contains the input data (for example, a saved search) that is passed when the <code>getInputData()</code> function is called.
	mapReduce.ReduceContext	Contains the key/values groups to process through the reduce stage. Passed in when <code>reduce(reduceContext)</code> is called.
	mapReduce.ReduceSummary	Holds statistics regarding the reduce stage.
	mapReduce.Summary	Holds statistics regarding the map/reduce process. Passed in when <code>summarize(summary)</code> is called.

GetInputContext Object Members

The following member is called on `mapReduce.GetInputContext`.

Member Type	Name	Return Type / Value Type	Description
Property	GetInputContext.isRestarted	read-only Boolean	Indicates whether <code>getInputData()</code> was stopped and invoked again.

InputSummary Object Members

The following members are called on the `mapReduce.InputSummary`.

Member Type	Name	Value Type	Description
Property	InputSummary.dateCreated	Date	The time and day when <code>getInputData()</code> began running.
	InputSummary.error	string	If applicable, holds a serialized error that is thrown from <code>getInputData()</code> .
	InputSummary.seconds	number	Total seconds elapsed when running <code>getInputData()</code> (does not include idle time).
	InputSummary.usage	number	Total number of usage units consumed when running <code>getInputData()</code> .

Iterator Object Members

The following members are called on the `mapReduce.Iterator`.

Member Type	Name	Return Type	Description
Method	Iterator.iterator().each(iteratorFunction)	void	Takes a developer-defined function. This function is called one time per element in a collection.

MapContext Object Members

The following members are called on `mapReduce.MapContext`.

Member Type	Name	Return Type / Value Type	Description
Property	MapContext.isRestarted	read-only Boolean	Indicates whether <code>map(mapContext)</code> was stopped and invoked again.
	MapContext.key	string	The input key to process through the map stage.
	MapContext.value	string	The input value to process through the map stage.

Member Type	Name	Return Type / Value Type	Description
	MapContext.write(key, value)	void	Writes the map stage output as key/value pairs.

MapSummary Object Members

The following members are called on the `mapReduce.MapSummary`.

Member Type	Name	Value Type	Description
Property	MapSummary.concurrency	number	Maximum concurrency number when running <code>map(mapContext)</code> .
	MapSummary.dateCreated	Date	The time and day when <code>map(mapContext)</code> began running.
	MapSummary.errors	mapReduce.Iterator	If applicable, holds a serialized error that is thrown from <code>map(mapContext)</code> .
	MapSummary.keys	mapReduce.Iterator	Count of unique keys passed to the <code>map(mapContext)</code> function.
	MapSummary.seconds	number	Total seconds elapsed when running <code>map(mapContext)</code> .
	MapSummary.usage	number	Total number of usage units consumed when running <code>map(mapContext)</code> .
	MapSummary.yields	number	Total number yields while running <code>map(mapContext)</code> .

ObjectRef Object Members

The following members are called on the `mapReduce.ObjectRef`.

Member Type	Name	Return Type / Value Type	Description
Property	ObjectRef.id	string number	The internal ID or script ID of the object. For example, the saved search ID.
	ObjectRef.type	string	The object's SuiteScript type ID.

ReduceContext Object Members

The following members are called on the `mapReduce.ReduceContext`.

Member Type	Name	Return Type / Value Type	Description
Property	ReduceContext.isRestarted	read-only Boolean	Indicates whether <code>ReduceContext.isRestarted</code> was stopped and invoked again.
	ReduceContext.key	string	The input key to process through the reduce stage.
	ReduceContext.values	string[]	The input values to process through the reduce stage.
Method	ReduceContext.write(key, value)	void	Writes the reduce stage output as key/value pairs.

ReduceSummary Object Members

The following members are called on the `mapReduce.ReduceSummary`.

Member Type	Name	Value Type	Description
Property	ReduceSummary.concurrency	number	Maximum concurrency number when running <code>reduce(reduceContext)</code> .
	ReduceSummary.dateCreated	Date	The time and day when <code>reduce(reduceContext)</code> began running.
	ReduceSummary.errors	mapReduce.Iterator	If applicable, holds a serialized error that is thrown from <code>reduce(reduceContext)</code> .
	ReduceSummary.keys	mapReduce.Iterator	Count of unique keys passed to the <code>reduce(reduceContext)</code> function.
	ReduceSummary.seconds	number	Total seconds elapsed when running <code>reduce(reduceContext)</code> .
	ReduceSummary.usage	number	Total number of usage units consumed when running <code>reduce(reduceContext)</code> .
	ReduceSummary.yields	number	Total number of yields when running <code>reduce(reduceContext)</code> .

Summary Object Members

The following members are called on the `mapReduce.Summary`.

Member Type	Name	Value Type	Description
Property	Summary.concurrency	number	Maximum concurrency number when running the map/reduce script.

Member Type	Name	Value Type	Description
	Summary.dateCreated	Date	The time and day when the map/reduce script began running.
	Summary.inputSummary	mapReduce.InputSummary	Statistics regarding the input stage.
	Summary.isRestarted	read-only Boolean	Indicates whether summarize(summary) was stopped and invoked again.
	Summary.mapSummary	mapReduce.MapSummary	Statistics regarding the map stage
	Summary.output	mapReduce.Iterator	Iterator that provides keys and values written as output during the reduce stage.
	Summary.reduceSummary	mapReduce.ReduceSummary	Statistics regarding the reduce stage
	Summary.seconds	number	Total seconds elapsed when running the map/reduce script.
	Summary.usage	number	Total number of usage units consumed when running the map/reduce script.
	Summary.yields	number	Total number of yields when running the map/reduce script.

mapReduce.GetInputContext

Object Description This object holds the [GetInputContext.isRestarted](#) property.

Since Version 2016 Release 1

Syntax

For a complete script example, see [Map/Reduce Script Sample](#).

GetInputContext.isRestarted

Property Description	Indicates whether it is the first run or a restarted run of the getInputData() function. Normally, the getInput function is invoked one time only. However, following a server stop and restart, NetSuite automatically restarts the getInput function. If the value of this property is <code>true</code> , the current process invoked by getInputData() was restarted. You can use this information to help you write a more robust map/reduce script that is designed to continue interrupted work as necessary.
Type	read-only Boolean

Since	Version 2016 Release 1
-------	------------------------

Syntax

```
...
function getInputData(context)
{
    if (context.isRestarted)
    {
        log.debug('GET_INPUT isRestarted', 'YES');
    }
    else
    {
        log.debug('GET_INPUT isRestarted', 'NO');
    }

    var extractSearch = search.load({ id: 'customsearch35' });
    return extractSearch;
}
...
...
```

mapReduce.ObjectRef

Object Description	References the object that contains the input data. For example, a reference to a saved search. You can use getInputData() to return this object.
---------------------------	---

 **Note:** The only supported object types are `search` and `file`.

Since	Version 2015 Release 2
-------	------------------------

Syntax

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

```
...
{
    type: 'search',
    id: 1234 //search internal id
}
...
```

ObjectRef.id

Property Description	The internal ID or script ID of the object. For example, the saved search ID.
Type	string number

Since	Version 2015 Release 2
-------	------------------------

Syntax

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

```
...
{
  type: 'search',
  id: 1234 //search internal id
}
...
```

ObjectRef.type

Property Description	The object's SuiteScript type ID.
Type	string
Values	'search'
Since	Version 2015 Release 2

Syntax

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

```
...
{
  type: 'search',
  id: 1234 //search internal id
}
...
```

mapReduce.MapContext

Object Description Contains the key/value pairs to process through the map stage.

Since Version 2015 Release 2

Syntax

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

```
...
function map(context)
```

```
{
    for (var i = 0; context.value && i < context.value.length; i++)
        if (context.value[i] !== ' ' && !PUNCTUATION_REGEX.test(context.value[i]))
            context.write(context.value[i], 1);
}
...
}
```

MapContext.isRestarted

Property Description	Indicates whether the <code>map(mapContext)</code> function was invoked again. If the Java virtual machine (JVM) restarts, to reduce negative impact to map/reduce processing, NetSuite automatically restarts the current map function. Map data previously written by the incomplete function is deleted. If the value is <code>true</code> , the current process invoked by <code>map(mapContext)</code> was restarted. You can use this information to help you write a more robust map/reduce script that is designed to continue interrupted work as necessary.
Type	read-only Boolean
Since	Version 2016 Release 1

Syntax

```
...
function map(context) {
    if (context.isRestarted)
    {
        log.debug('MAP for ' + context.key + ' isRestarted', 'YES');
    }
    else
    {
        log.debug('MAP for ' + context.key + ' isRestarted', 'NO');
    }
    context.write(context.key, context.value);
...
}
```

MapContext.key

Property Description	The key to be processed through the map stage. <ul style="list-style-type: none"> ■ If the input type is an Array, the key is the index of the element. ■ If the input type is an Object, the key is the key in the Object. ■ If the input type is a result set, the key is the internal ID of the result. If the search result has no internal ID, the key is the index of the search result. <p>Note: Each key cannot exceed 4000 bytes.</p>
Type	string
Since	Version 2015 Release 2

Syntax

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

```
...
map: function (context)
{
    // for a search.Search input, context.key is Internal ID
    context.write(record.transform({
        fromType: record.Type.SALES_ORDER,
        toType: record.Type.ITEM_FULFILLMENT,
        fromId: context.key
    }))
...
}
```

MapContext.value

Property Description	The value to be processed through the map stage. <ul style="list-style-type: none"> ■ If the input type is an Array, it is the value in the element. ■ If the input type is an Object, it is the value in the Object. ■ If the input type is a result set, the value is a <code>search.Result</code> object converted to a JSON string with <code>JSON.stringify()</code>.
Type	string
Since	Version 2015 Release 2

Syntax

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

```
...
map: function (context)
{
    // for search.Search input, context.value is a search.SearchResult
    context.write(record.transform({
        fromType: record.Type.SALES_ORDER,
        toType: record.Type.ITEM_FULFILLMENT,
        fromId: context.key
    }))
...
}
```

MapContext.write(key,value)

Method Description	Writes the key/value pairs.
---------------------------	-----------------------------

Returns	Void
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description
key	string	required	The key to write
value	any	required	The value to write

Syntax

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

```
...
function map(context)
{
    for (var i = 0; context.value && i < context.value.length; i++)
        if (context.value[i] !== ' ' && !PUNCTUATION_REGEX.test(context.value[i]))
            context.write(context.value[i], 1);
}
...
```

mapReduce.ReduceContext

Object Description	Contains the key/values groups to process through the reduce stage.
---------------------------	---

Since	Version 2015 Release 2
--------------	------------------------

Syntax

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

```
...
function reduce(context)
{
    context.write(context.key, context.values.length);
}
...
```

ReduceContext.isRestarted

Property Description	Indicates whether the <code>reduce(reduceContext)</code> function was invoked again.
-----------------------------	--

	If the Java virtual machine (JVM) restarts, to reduce negative impact to map/reduce processing, NetSuite automatically restarts the current reduce function. Reduce data previously written by the incomplete function is deleted. If the value is <code>true</code> , the current process invoked by <code>reduce(reduceContext)</code> was restarted. You can use this information to help you write a more robust map/reduce script that is designed to continue interrupted work as necessary.
Type	read-only Boolean
Since	Version 2016 Release 1

Syntax

```
...
function reduce(context) {
  if (context.isRestarted)
  {
    log.debug('REDUCE for ' + context.key + ' isRestarted', 'YES');
  }
  else
  {
    log.debug('REDUCE for ' + context.key + ' isRestarted', 'NO');
  }
  context.write(context.key, context.values.length);
...
}
```

ReduceContext.key

Property Description	<p>When the map/reduce process includes a map stage, the key is derived from the key written by <code>MapContext.write(key,value)</code>. When the map stage is skipped, the key depends on the input type:</p> <ul style="list-style-type: none"> ■ If the input type is an Array, the key is the index of the element. ■ If the input type is an Object, the key is the key in the Object. ■ If the input type is a result set, the key is the internal ID of the result. <p>Note: Each key cannot exceed 4000 bytes.</p>
Type	string
Since	Version 2015 Release 2

Syntax

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

```
...
function reduce(context)
{
  context.write(context.key, context.values.length);
}
...
```

ReduceContext.values

Property Description	<p>When the map/reduce process includes a map stage, the values are derived from the values written by MapContext.write(key,value). When the map stage is skipped, the values are already grouped by key into a list, and the value depends on the input type:</p> <ul style="list-style-type: none"> ■ If the input type is an Array, it is the value in the element. ■ If the input type is an Object, it is the value in the Object. ■ If the input type is a result set, the value is a <code>search.Result</code> object converted to a JSON string with <code>JSON.stringify()</code>. <p>Note: Each value cannot exceed 1 megabyte.</p>
Type	<code>string[]</code>
Since	Version 2015 Release 2

Syntax

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

```
...
function reduce(context)
{
    context.write(context.key, context.values.length);
}
...
```

ReduceContext.write(key, value)

Method Description	Writes the key/values groups.
Returns	Void
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description
key	<code>string</code>	required	The key to write.
value	any	required	The value to write

Syntax

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

```
...
function reduce(context)
```

```
{
    context.write(context.key, context.values.length);
}
...
}
```

mapReduce.Iterator

Object Description	Provides the keys and values written as output during the reduce stage.
Since	Version 2015 Release 2

Syntax

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

```
...
summary.output.iterator().each(function (key, value)
{
    contents += (key + ' ' + value + '\n');
    return true;
});
...
}
```

Iterator.iterator().each(iteratorFunction)

Method Description	Method that takes a developer-defined function. This function is called one time per element in a collection. The iteratorfunction holds the custom logic that you want to execute on each element in your collection of data. For each call to iterator(), a new iterator Object is used.
Returns	Void
Since	Version 2016 Release 1

Parameters

Parameter	Type	Required / Optional	Description
IteratorFunction	function	required	A function that is executed on each key/value pair.
IteratorFunction.key	string	required	Callback parameter The keys to iterate on.  Note: Each key cannot exceed 4000 bytes.
IteratorFunction.value	string	required	Callback parameter The values to iterate on.

Parameter	Type	Required / Optional	Description
			<p>Note: Each value cannot exceed 1 megabyte.</p>

Syntax

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

```
...
summary.output.iterator().each(function (key, value)
{
    contents += (key + ' ' + value + '\n');
    return true;
});
...
```

mapReduce.Summary

Object Description	Holds statistics regarding execution of a map/reduce script.
Since	Version 2015 Release 2

Syntax

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

```
...
function summarize(summary)
{
    var type = summary.toString();
    log.audit(type + ' Usage Consumed', summary.usage);
    log.audit(type + ' Concurrency Number ', summary.concurrency);
    log.audit(type + ' Number of Yields', summary.yields);
    var contents = '';
    summary.output.iterator().each(function (key, value)
    {
        contents += (key + ' ' + value + '\n');
        return true;
    });
...
}
```

Summary.dateCreated

Property Description	The time and day when the map/reduce script began running.
Type	Date

Since	Version 2015 Release 2
-------	------------------------

Syntax

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

```
...
log.audit(type + ' Creation Date', summary.dateCreated);
...
```

Summary.seconds

Property Description	Total seconds elapsed when running the map/reduce script.
Type	number
Since	Version 2015 Release 2

Syntax

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

```
...
log.audit(type + ' Total seconds elapsed', summary.seconds);
...
```

Summary.usage

Property Description	Total number of usage units consumed when running the map/reduce script.
Type	number
Since	Version 2015 Release 2

Syntax

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

```
...
log.audit(type + ' Usage Consumed', summary.usage);
...
```

Summary.concurrency

Property Description	The maximum concurrency number when executing parallel tasks for the map/reduce script.
----------------------	---

	<p>Note: This number may be less than the number allocated on the script deployment. For example, tasks that remained in the pending state are not reflected in this number.</p>
Type	number
Since	Version 2015 Release 2

Syntax

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

```
...
log.audit(type + ' Concurrency Number ', summary.concurrency);
...
```

Summary.yields

Property Description	Total number of yields when running the map/reduce script.
Type	number
Since	Version 2015 Release 2

Syntax

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

```
...
log.audit(type + ' Number of Yields', summary.yields);
...
```

Summary.inputSummary

Property Description	Statistics regarding the input stage.
Type	mapReduce.InputSummary
Since	Version 2015 Release 2

Syntax

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

```
...
logMetrics(summary.inputSummary);
```

```
log.error('Input Error', summary.inputSummary.error);
...
```

Summary.mapSummary

Property Description	Statistics regarding the map stage.
Type	mapReduce.MapSummary
Since	Version 2015 Release 2

Syntax

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

```
...
logMetrics(summary.mapSummary);
var mapKeys = [];
summary.mapSummary.keys.iterator().each(function (key)
{
    mapKeys.push(key);
    return true;
});
log.audit('MAP keys processed', mapKeys);
summary.mapSummary.errors.iterator().each(function (key, error)
{
    log.error('Map Error for key: ' + key, error);
    return true;
});
...
...
```

Summary.reduceSummary

Property Description	Statistics regarding the reduce stage.
Type	mapReduce.ReduceSummary
Since	Version 2015 Release 2

Syntax

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

```
...
summary.reduceSummary.errors.iterator().each(function (key, error)
{
    log.error('Reduce Error for key: ' + key, error);
    return true;
});
```

```
    });
...
}
```

Summary.output

Property Description	Iterator that provides keys and values written as output during the reduce stage.
Type	mapReduce.Iterator
Since	Version 2015 Release 2

Syntax

```
...
summary.output.iterator().each(function (key, value)
{
    contents += (key + ' ' + value + '\n');
    return true;
});
...
}
```

Summary.isRestarted

Property Description	Indicates whether the summarize(summary) function was invoked again. To reduce negative impact to map/reduce processing if the Java virtual machine (JVM) restarts, NetSuite automatically restarts the current summary function. Summary data previously written by the incomplete function is deleted. If the value is <code>true</code> , the current process invoked by summarize(summary) was restarted. You can use this information to help you write a more robust map/reduce script that is designed to continue interrupted work as necessary.
Type	read-only Boolean
Since	Version 2016 Release 1

Syntax

```
...
function summarize(summary) {
    if (summary.isRestarted)
    {
        log.debug('SUMMARY isRestarted', 'YES');
    }
    else
    {
        log.debug('SUMMARY isRestarted', 'NO');
    }
    log.debug('summarize', JSON.stringify(summary));
}
...
```

mapReduce.InputSummary

Object Description	Holds statistics regarding the input stage.
Since	Version 2015 Release 2

Syntax

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

```
...
logMetrics(summary.inputSummary);
log.error('Input Error', summary.inputSummary.error);

...
```

InputSummary.dateCreated

Property Description	The time and day when <code>getInputData()</code> began running.
Type	Date
Since	Version 2015 Release 2

Syntax

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

```
...
log.audit(' Creation Date', summary.inputSummary.dateCreated);
...
```

InputSummary.seconds

Property Description	Total seconds elapsed when running <code>getInputData()</code> (does not include idle time).
Type	number
Since	Version 2015 Release 2

Syntax

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

```
...
log.audit(' Time Elapsed', summary.inputSummary.seconds);
...
```

InputSummary.usage

Property Description	Total number of usage units consumed when running <code>getInputData()</code> .
Type	number
Since	Version 2015 Release 2

Syntax

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

```
...
log.audit(' Usage', summary.inputSummary.usage);
...
```

InputSummary.error

Property Description	If applicable, holds a serialized error that is thrown from <code>getInputData()</code> .
Type	string
Since	Version 2015 Release 2

Syntax

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

```
...
logMetrics(summary.inputSummary);
log.error('Input Error', summary.inputSummary.error);

...
```

mapReduce.MapSummary

Object Description	Holds statistics regarding the map stage
Since	Version 2015 Release 2

Syntax

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

```
...
logMetrics(summary.mapSummary);
var mapKeys = [];
```

```

summary.mapSummary.keys.iterator().each(function (key)
{
    mapKeys.push(key);
    return true;
});
log.audit('MAP keys processed', mapKeys);
summary.mapSummary.errors.iterator().each(function (key, error)
{
    log.error('Map Error for key: ' + key, error);
    return true;
});
...

```

MapSummary.dateCreated

Property Description	The time and day when <code>map(mapContext)</code> began running.
Type	Date
Since	Version 2015 Release 2

Syntax

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

```

...
log.audit(' Creation Date', summary.mapSummary.dateCreated);
...
```

MapSummary.seconds

Property Description	Total seconds elapsed when running <code>map(mapContext)</code> .
Type	number
Since	Version 2015 Release 2

Syntax

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

```

...
log.audit(' Time Elapsed', summary.mapSummary.seconds);
...
```

MapSummary.usage

Property Description	Total number of usage units consumed when running <code>map(mapContext)</code> .
-----------------------------	--

Type	number
Since	Version 2015 Release 2

Syntax

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

```
...
log.audit(' Usage', summary.mapSummary.usage);
...
```

MapSummary.concurrency

Property Description	The maximum concurrency number for executing parallel tasks during the map stage.
	Note: This number may be less than the concurrency number allocated on the script deployment. For example, tasks that remained in the pending state are not reflected in this number.
Type	number
Since	Version 2015 Release 2

Syntax

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

```
...
log.audit(' Concurrency', summary.mapSummary.concurrency);
...
```

MapSummary.yields

Property Description	Total number of times yields when running <code>map(mapContext)</code> .
Type	number
Since	Version 2015 Release 2

Syntax

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

```
...
log.audit(' Total Yields', summary.mapSummary.yields);
```

...

MapSummary.keys

Property Description	Count of unique keys passed to the map(mapContext) function.
Type	mapReduce.Iterator
Since	Version 2015 Release 2

Syntax

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

```
...
logMetrics(summary.mapSummary);
var mapKeys = [];
summary.mapSummary.keys.iterator().each(function (key)
{
    mapKeys.push(key);
    return true;
});
...
```

MapSummary.errors

Property Description	If applicable, holds a serialized error that is thrown from map(mapContext).
Type	mapReduce.Iterator
Since	Version 2015 Release 2

Syntax

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

```
...
summary.mapSummary.errors.iterator().each(function (key, error)
{
    log.error('Map Error for key: ' + key, error);
    return true;
});
...
```

mapReduce.ReduceSummary

Object Description	Holds statistics regarding the reduce stage.
---------------------------	--

Since	Version 2015 Release 2
-------	------------------------

Syntax

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

```
...
summary.reduceSummary.errors.iterator().each(function (key, error)
{
    log.error('Reduce Error for key: ' + key, error);
    return true;
});
...
```

ReduceSummary.dateCreated

Property Description	The time and day when <code>reduce(reduceContext)</code> began running.
Type	Date
Since	Version 2015 Release 2

Syntax

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

```
...
log.audit(' Creation Date', summary.reduceSummary.dateCreated);
...
```

ReduceSummary.seconds

Property Description	Total seconds elapsed when running <code>reduce(reduceContext)</code> .
Type	number
Since	Version 2015 Release 2

Syntax

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

```
...
log.audit(' Time Elapsed', summary.reduceSummary.seconds);
```

...

ReduceSummary.usage

Property Description	Total number of usage units consumed when running <code>reduce(reduceContext)</code> .
Type	number
Since	Version 2015 Release 2

Syntax

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

```
...
log.audit(' Usage', summary.mapSummary.usage);
...
```

ReduceSummary.concurrency

Property Description	The maximum concurrency number for executing parallel tasks during the reduce stage.
	Note: This number may be less than the concurrency number allocated on the script deployment. For example, tasks that remained in the pending state are not reflected in this number.
Type	number
Since	Version 2015 Release 2

Syntax

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

```
...
log.audit(' Concurrency', summary.reduceSummary.concurrency);
...
```

ReduceSummary.yields

Property Description	Total number of times yields when running <code>reduce(reduceContext)</code> .
Type	number
Since	Version 2015 Release 2

Syntax

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

```
...
log.audit(' Total Yields', summary.reduceSummary.yields);
...
```

ReduceSummary.keys

Property Description	Count of unique keys passed to the map(mapContext) function.
Type	mapReduce.Iterator
Since	Version 2015 Release 2

Syntax

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

```
...
var reduceKeys = [];
summary.reduceSummary.keys.iterator().each(function (key)
{
    reduceKeys.push(key);
    return true;
});
log.audit('REDUCE keys processed', reduceKeys);
...
```

ReduceSummary.errors

Property Description	If applicable, holds a serialized error that is thrown from reduce(reduceContext).
Type	mapReduce.Iterator
Since	Version 2015 Release 2

Syntax

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

```
...
summary.reduceSummary.errors.iterator().each(function (key, error)
{
    log.error('Reduce Error for key: ' + key, error);
    return true;
});
...
```

Mass Update Script Type

Mass update scripts allow you to programmatically perform custom updates to fields that are not available through general mass updates. Mass update scripts can run complex calculations, as defined in your script, across records.

Mass Update scripts are executed on the server when you click the Perform Update button on the Mass Update Preview Results page.

Mass Update Script Sample

The following code updates the probability field of all existing records to 61%.

Note: From the script deployment record, ensure that the **Applies To** field is set to Opportunity.

```
/** 
 * @NApiVersion 2.0
 * @NScriptType MassUpdateScript
 */
define(['N/record'],
    function(record) {
        function each(params) {
            // Set the probability to 61%
            var recOpportunity = record.load({
                type: params.type,
                id: params.id
            });
            recOpportunity.setValue('probability', 61);
            recOpportunity.save();
        }
        return {
            each: each
        };
    });
});
```

Mass Update Script Entry Points

Script Entry Point	
each	Iterates through each applicable record.

each

Description	Iterates through each applicable record.
Returns	void
Since	Version 2016 Release 1

Parameters

Note: The `params` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
<code>params.id</code>	number	The ID of the record being processed by the mass update.	Version 2016 Release 1
<code>params.type</code>	string	The record type of the record being processed by the mass update.	Version 2016 Release 1

Portlet Script Type

Portlet scripts are run on the server and are rendered in the NetSuite dashboard.

The following portlet script types are supported:

- **FORM** — A data entry form with up to one submit button embedded into a portlet. This type supports the Portlet module that can refresh and resize the portlet, as well as the use of record-level client-side script to implement validation. See [N/portlet Module](#).
- **HTML** — An HTML-based portlet that is used to display free-form HTML. (images, Flash, custom HTML)
- **LINKS** — A portlet that consists of rows of formatted content.
- **LIST** — A standard list of user-defined column headers and rows.

You can designate that a portlet script should be used for a SuiteApp portlet. A SuiteApp portlet is a specialized type of custom portlet. It provides direct access from users' dashboards to a SuiteApp installed in their account. This type of script is called a Dashboard SuiteApp portlet script. If a Dashboard SuiteApp portlet is included in a SuiteApp, a user can add a SuiteApp portlet to their dashboard from the Personalize Dashboard menu. For instructions, see the help topic [SuiteApp Portlets](#). This type of portlet is supported for installed SuiteApps that include a dashboard component. A SuiteApp portlet script not only provides content for SuiteApp portlets. It also enables you to include your choice of graphics as branding for the icons that are shown for SuiteApp portlets in the Personalize Dashboard window.

To view content produced by a portlet script that is not intended for a SuiteApp portlet, a user must add a custom portlet to their dashboard and select the script in the portlet setup. For more information, see the help topics [Custom Portlets](#) and [Adding a Portlet to a Dashboard](#).

For steps to create and deploy a portlet script, see [Creating and Deploying a Portlet Script](#).

Creating and Deploying a Portlet Script

The following is a basic process flow for creating and deploying a portlet script, including a Dashboard SuiteApp portlet script:

1. Create a portlet script and upload the file to your File Cabinet.
For more information about this process, see [SuiteScript 2.0 – Script Creation Process](#).
2. Create a script record for your portlet script, and ensure that the **Portlet Type** field is set to the correct portlet type used in your script.
For more information about this process, see [Creating a Script Record](#).

3. Create a script deployment record for your portlet script.

If you are creating a script to be used for a SuiteApp portlet, enable the **Dashboard SuiteApp** option.

When this option is enabled, you can upload an image to the file cabinet. This image represents the SuiteApp icon shown for the portlet in the Personalize Dashboard window. Note that only SVG images are supported for the icon. SVG is a vector format, so it assures perfect image scalability. For more information about icon guidelines, see [Guidelines for Creating a Dashboard SuiteApp Icon](#).

The following image indicates where the Dashboard fields are located:

The screenshot shows the 'Script Deployment' dialog box. At the top are 'Save' and 'Cancel' buttons. Below is a 'SCRIPT' section with the value 'Financial Report Systems Form Portlet'. Under 'TITLE *' is a text input field containing 'Financial Report Systems Form Portlet'. The 'ID' field is empty. The 'DEPLOYED' checkbox is checked. The 'DASHBOARD SUITEAPP' checkbox is checked and is highlighted with a red border. The 'ICON' section shows a dropdown menu with 'NetSuite Logo' selected.

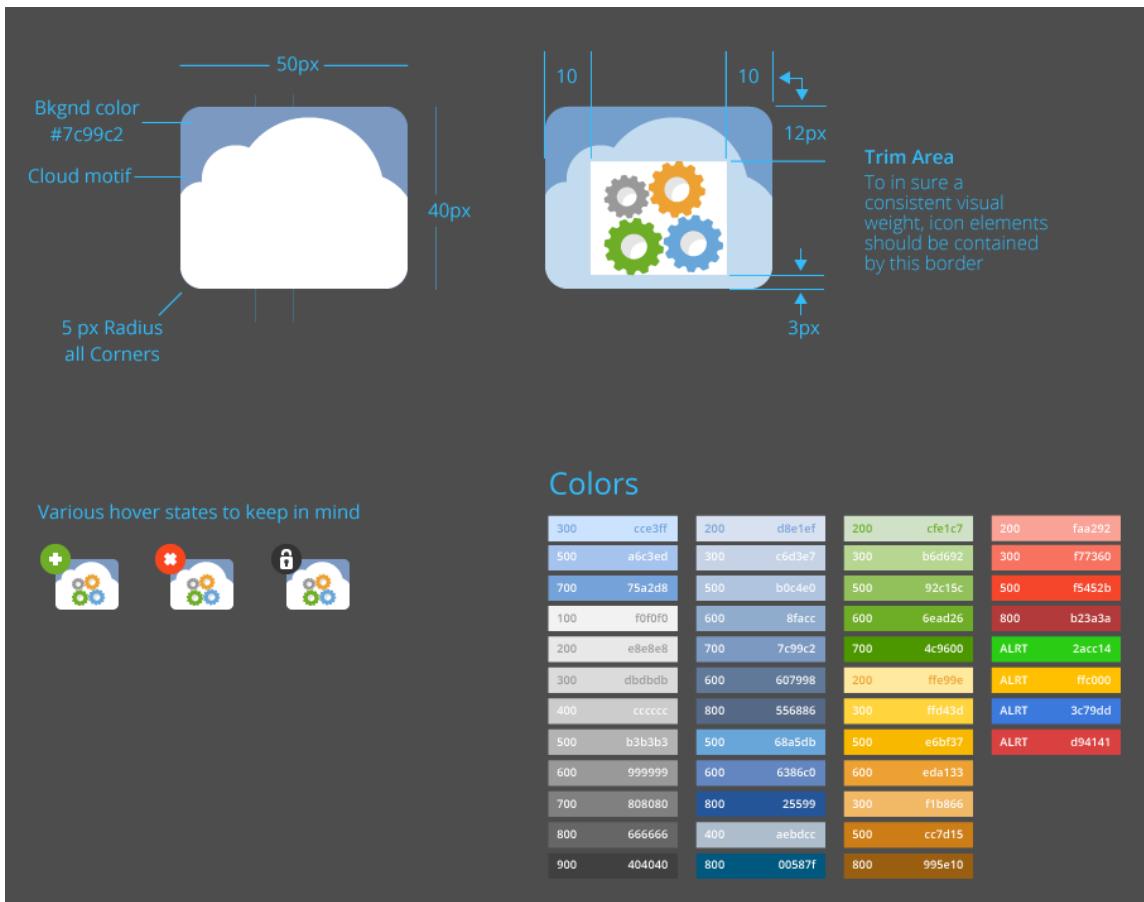
For more information about the script deployment process, see [Methods of Deploying a Script](#).

Guidelines for Creating a Dashboard SuiteApp Icon

Dashboard SuiteApp icons require certain visual characteristics to align them with other icons in the dashboard. The following guidelines preserve the visual characteristics of Dashboard SuiteApp icons:

- Dashboard SuiteApp icons use a unique background with a cloud motif. This background ensures a common size and shape for all Dashboard SuiteApp icons and provides a cloud motif that serves as an additional modifier.
- NetSuite recommends using a restricted color palette. This palette is a major contributor to the character of the icons used in NetSuite. Restrict your colors to this palette whenever possible. If additional colors are required, add them as special exceptions. See the image below for more information about the recommended color palette.
- Graphical elements should be more geometrical than illustrative. Avoid using complex and irregular shapes. Try to reduce elements in your composition to their most primitive geometry.
- Dashboard SuiteApp icons use a simulated light source to increase detail and definition in icon elements. This light source is evidenced by a cast shadow that proceeds down and to the right of elements in the composition at a 45 degree angle. Cast shadows are rendered with a black fill set to 10% opacity.
- Your icon image must be saved as an SVG file.

The following image shows the guidelines in use:



Form Portlet Script Sample

```
/**
 * @NApiVersion 2.x
 * @NScriptType Portlet
 */
define(['N/search'],
    function(search) {
        function render(params) {
            var portlet = params.portlet;
            portlet.title = 'Simple Form Portlet';
            var fld = portlet.addField({
                id: 'text',
                type: 'text',
                label: 'Text'
            });
            fld.updateLayoutType({
                layoutType: 'normal'
            });
            fld.updateBreakType({
                breakType: 'startcol'
            });
            portlet.setSubmitButton({
                url: 'http://httpbin.org/post',
            })
        }
    }
);
```

```

        label: 'Submit',
        target: '_top'
    });
}
return {
    render: render
};
});

```

HTML Portlet Script Sample

```

/**
 *@NApiVersion 2.x
 *@NScriptType Portlet
 */
define([], 
    function() {
        function render(params) {
            params.portlet.title = 'My Portlet';
            var content = '<td><span><b>Hello!!!</b></span></td>';
            params.portlet.html = content;
        }
        return {
            render: render
        };
    });

```

Links Portlet Script Sample

```

/**
 *@NApiVersion 2.x
 *@NScriptType Portlet
 */
define(['N/search'],
    function(search) {
        function render(params) {
            var portlet = params.portlet;
            portlet.title = 'Search Engines';
            portlet.addLine({
                text: 'Google',
                url: 'http://www.google.com/'
            });
            portlet.addLine({
                text: 'Bing',
                url: 'http://www.bing.com/'
            });
        }
        return {
            render: render
        };
    });

```

List Portlet Script Sample



Important: This sample references a custom entity field with the ID `custentity_multiselect`. Before attempting to use this sample, you must either create a field with that ID or edit the sample so that it references an existing custom entity field in your account.

```
/**  
 * @NApiVersion 2.x  
 * @NScriptType Portlet  
 */  
define(['N/search'],  
    function(search) {  
        function render(params) {  
            var isDetail = (params.column === 2);  
            var portlet = params.portlet;  
            portlet.title = isDetail ? "My Detailed List" : "My List";  
            portlet.addColumn({  
                id: 'internalid',  
                type: 'text',  
                label: 'Number',  
                align: 'LEFT'  
            });  
            portlet.addColumn({  
                id: 'entityid',  
                type: 'text',  
                label: 'ID',  
                align: 'LEFT'  
            });  
            if (isDetail) {  
                portlet.addColumn({  
                    id: 'email',  
                    type: 'text',  
                    label: 'E-mail',  
                    align: 'LEFT'  
                });  
                portlet.addColumn({  
                    id: 'custentity_multiselect',  
                    type: 'text',  
                    label: 'Multiselect',  
                    align: 'LEFT'  
                });  
            }  
            var filter = search.createFilter({  
                name: 'email',  
                operator: search.Operator.ISNOTEMPTY  
            });  
            var customerSearch = search.create({  
                type: 'customer',  
                filters: filter,  
                columns: ['internalid', 'entityid', 'email', 'custentity_multiselect']  
            });  
            var count = isDetail ? 15 : 5;  
            customerSearch.run().each(function(result) {
```

```

        portlet.addRow(result.getAllValues());
        return --count > 0;
    });
}
return {
    render: render
};
});

```

render(params)

Description	Definition of the portlet script trigger point.
Returns	void
Since	Version 2015 Release 2

Parameters

 Note:	The <code>params</code> parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.
--	---

Parameter	Type	Description	Since
<code>params.portlet</code>	Portlet Object	The Portlet object used for rendering. The availability of Portlet members depends on the type of Portlet that is passed in. For more information, see the following: <ul style="list-style-type: none">▪ Form Portlet Object Members▪ HTML Portlet Object Members▪ Links Portlet Object Members▪ List Portlet Object Members	Version 2015 Release 2
<code>params.column</code>	integer	The column index for the portlet on the dashboard. The index is represented by one of the following numeric values: <ol style="list-style-type: none">1. left column2. center column3. right column	Version 2015 Release 2
<code>params.entityid</code>	string	The customer ID for the selected customer.	Version 2015 Release 2

Portlet Object

Portlet objects are used to encapsulate scriptable dashboard portlets. They are automatically passed to the `render(params)` entry point by NetSuite. For more information, see [render\(params\)](#).

Form Portlet Object Members

These members are only available to Form Portlet objects.

Member Type	Name	Return Type / Value Type	Description
Method	Portlet.addField(options)	serverWidget.Field	Adds a field to the form.
	Portlet.setSubmitButton(options)	serverWidget.Button	Adds a submit button to the form.
Property	Portlet.clientScriptFileId	number	The script file ID to be used in the portlet.
	Portlet.clientScriptModule Path	string	The script path to be used in the portlet.
	Portlet.title	string	The title of the portlet.

HTML Portlet Object Members

These members are only available to HTML Portlet objects.

Member Type	Name	Return Type / Value Type	Description
Property	Portlet.html	string	The complete HTML contents of the portlet.
	Portlet.title	string	The title of the portlet.

Links Portlet Object Members

These members are only available to Links Portlet objects.

Member Type	Name	Return Type / Value Type	Description
Method	Portlet.addLine(options)	Object	Adds a line to the portlet.
Property	Portlet.title	string	The title of the portlet.

List Portlet Object Members

These members are only available to List Portlet objects.

Member Type	Name	Return Type / Value Type	Description
Method	Portlet.addColumn(options)	serverWidget.ListColumn	Adds a column to the portlet.
	Portlet.addEditColumn(options)	serverWidget.ListColumn	Adds an Edit or Edit/View column to the portlet.
	Portlet.addRow(options)	Object	Adds a row to the portlet.

Member Type	Name	Return Type / Value Type	Description
	Portlet.addColumn(options)	Object	Adds multiple rows to the portlet.
Property	Portlet.title	string	The title of the portlet.

Portlet.addColumn(options)

Method Description	Adds a list column to the portlet.
Returns	serverWidget.ListColumn
Entry Point	render(params)
Since	2016.2

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.	2016.2
options.label	string	required	The label of this column.	2016.2
options.type	string	required	The field type for this column. For more information about possible values, see serverWidget.FieldType.	2016.2
options.align	string	optional	The layout justification for this column. For more information about possible values, see serverWidget.LayoutJustification.	2016.2

Syntax

The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see Portlet Script Type.

```
...
var newColumn = params.portlet.addColumn({
    id: 'column1',
    label: 'Text',
    type: serverWidget.FieldType.TEXT,
    align: serverWidget.LayoutJustification.RIGHT
});
...
```

Portlet.addEditColumn(options)

Method Description	Adds an Edit or Edit/View column to the portlet.
Returns	serverWidget.ListColumn
Entry Point	render(params)
Since	2016.2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.column	string	required	The internal ID of the column to the left of which the Edit/View column is added.	2016.2
options.showHrefCol	boolean true false	optional	If true, the URL for the link is clickable. The default setting is false.	2016.2
options.showView	boolean true false	optional	If true, then an Edit/View column is added. Otherwise, only an Edit column is added. The default setting is false.	2016.2

Syntax

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

```
...
var newColumn = params.portlet.addEditColumn({
    column: 'column1',
    showHrefCol: true,
    showView: true
});
```

Portlet.addField(options)

Method Description	Adds a field to the form.
Returns	serverWidget.Field

Entry Point	render(params)
Since	2016.2

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 <code>custpage</code> if you are adding the field to an existing page. For example, if you add a field that appears as Purchase Details , the field internal ID should be something similar to <code>custpage_purchasedetails</code> or <code>custpage_purchase_details</code> .	2016.2
options.label	string	required	The label for this field.	2016.2
options.type	string	required	The field type for the field. For more information about possible values, see serverWidget.FieldType .	2016.2
options.source	string	optional	<p>The internal ID or script ID of the source list for this field if it is a select (List/Record) or multi-select field.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">  Note: For radio fields only, the <code>source</code> parameter must contain the internal ID for the field. </div> <div style="border: 1px solid #ccc; background-color: #ffffcc; padding: 10px; margin-top: 10px;">  Important: 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. </div> <p>For more information about working with radio buttons, see the help topic Working with Radio Buttons.</p>	2016.2

Syntax

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

```
...
var newField = params.portlet.addField({
    id: 'textfield',
    type: serverWidget.FieldType.TEXT,
    label: 'text'
});
...
```

Portlet.addLine(options)

Method Description	Adds a line to the portlet.
Returns	Object
Entry Point	render(params)
Since	2016.2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.text	string	required	The text for the line.	2016.2
options.url	string	optional	The URL link.	2016.2
options.align	number	optional	The justification for the line. This value indicates the number of spaces to indent.	2016.2

Syntax

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

```
...
params.portlet.addLine({
    text: 'Google',
    url: 'http://www.google.com',
    align: 4
});
```

...

Portlet.addRow(options)

Method Description	Adds a row to the portlet.
Returns	Object
Entry Point	render(params)
Since	2016.2

Parameters

i Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.row	search.Result string	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.	2016.2

Syntax

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

```
...
params.portlet.addRow({
    row: {
        columnid1: 'value1',
        columnid2: 'value2'
    }
});
...
```

Portlet.addRows(options)

Method Description	Adds multiple rows to the portlet.
Returns	Object
Entry Point	render(params)
Since	2016.2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.rows	search.Result[] string[]	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.	2016.2

Syntax

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

```
...
params.portlet.addRows({
  rows:
    [
      {
        columnid1: 'value1',
        columnid2: 'value2'
      }, {
        columnid1: 'value2',
        columnid2: 'value3'
      }
    ]
});
...
...
```

Portlet.setSubmitButton(options)

Method Description	Adds a submit button to the form.
Returns	serverWidget.Button
Entry Point	render(params)
Since	2016.2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.url	string	required	The URL that the form posts data to.	2016.2

Parameter	Type	Required / Optional	Description	Since
options.label	string	optional	The button label.	2016.2
options.target	string	optional	The target attribute of the form element, if it is different from the portlet's own embedded iframe. Supported values include standard HTML target attributes such as <code>_top</code> , <code>_parent</code> , and <code>_blank</code> . It can also be set to frame names and the NetSuite-specific identifier <code>_hidden</code> . Setting this value to <code>_hidden</code> allows submission to a backend that returns results to a hidden child iframe within the portlet's embedded iframe.	2016.2

Syntax

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

```
...
params.portlet.setSubmitButton({
    url: 'http://httpbin.org/post',
    label: 'Submit',
    target: '_top'
});
...
```

Portlet.clientScriptFileId

Property Description	The script file ID to be used in the portlet.
Type	number
Entry Point	<code>render(params)</code>
Since	2016.2

Errors

Error Code	Thrown If
PROPERTY_VALUE_CONFLICT	You attempted to set this value when the <code>Portlet.clientScriptModulePath</code> property value has already been specified. For more information, see Portlet.clientScriptFileId .

Syntax

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

```
...
params.portlet.clientScriptModulePath = '/SuiteScripts/clientScript.js';
...
```

Portlet.clientScriptModulePath

Property Description	The script path to be used in the portlet.
Type	string
Entry Point	render(params)
Since	2016.2

Errors

Error Code	Thrown If
PROPERTY_VALUE_CONFLICT	You attempted to set this value when the Portlet.clientScriptModulePath property value has already been specified. For more information, see Portlet.clientScriptModulePath .

Syntax

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

```
...
params.portlet.clientScriptModulePath = '/SuiteScripts/clientScript.js';
...
```

Portlet.html

Property Description	The complete HTML contents of the portlet.
Type	string
Entry Point	render(params)
Since	2016.2

Syntax

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

```
...
params.portlet.html = htmlcontents;
...
```

Portlet.title

Property Description	The title of the portlet.
Type	string
Entry Point	render(params)
Since	2016.2

Syntax

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

```
...
params.portlet.title = 'My Portlet';
...
```

RESTlet Script Type

A RESTlet is a SuiteScript that you make available for other applications to call, either from an external application or from within NetSuite. When an application or another script calls a RESTlet, the RESTlet script executes and, in some cases, returns a value to the calling application.

RESTlets can be useful when you want to bring data from another system into NetSuite, or if you want to extract data from NetSuite. RESTlets can also be used, in combination with other scripts, to customize the behavior of a page within NetSuite.

For more details, see the following sections:

- [Getting Started with RESTlets](#)
- [RESTlet Authentication](#)
- [RESTlet Reference](#)
- [RESTlet Script and Request Samples](#)
- [RESTlet Script Entry Points](#)

Getting Started with RESTlets

For help getting started with RESTlets, see the following topics:

- [RESTlet Key Concepts](#)
- [Deploying a RESTlet](#)
- [Identifying a RESTlet in a Call](#)
- [Selecting an HTTP Method for Calling a RESTlet](#)
- [Creating a Content-Type Header](#)

RESTlet Key Concepts

A RESTlet is a SuiteScript that executes when called by an external application or by another SuiteScript. Depending on how the RESTlet is written and called, it may also return data to the calling application.

As with other script types, RESTlets have broad potential applications. A RESTlet can perform any function that can be implemented by using SuiteScript. But at a high level, potential uses of RESTlets generally include the following:

- Retrieving, adding, or manipulating data within NetSuite, from an external source. In this sense, RESTlets can be seen as an alternative to NetSuite's SOAP-based web services.
- Customizing the behavior of pages and features within NetSuite. In this sense, RESTlets can be seen as an alternative to other script types, such as server-side Suitelets. The advantage of using a RESTlet compared with a Suitelet is that the RESTlet can return data, in plain text or JSON, to the client script.

To use a RESTlet, you follow the same guidelines as you would with an entry point script deployed at the record level. For example, you must create a script record and a deployment record based on the RESTlet script file. These processes are described further in [Deploying a RESTlet](#).

When you save a script deployment record for a RESTlet, the system automatically generates a URL that can be used to call the RESTlet. Because a RESTlet executes only when it is called, this information is critical for using the RESTlet. For more details, see [Identifying a RESTlet in a Call](#).

When you are ready to call a RESTlet that you have deployed, you can use one of four supported HTTP methods: delete, get, post, or put. Depending on which method you use, you may be required to embed input for the RESTlet in the URL, or you may be required to submit arguments in a request body. Additionally, for the call to be successful, your RESTlet script must contain an entry point that corresponds with the method you use to make the call. For details on supported HTTP methods and formatting your request, see [Selecting an HTTP Method for Calling a RESTlet](#).

One advantage of RESTlets over other script types is that NetSuite requires authentication for RESTlets. If a RESTlet call originates from a client that does not have an existing session in the NetSuite account where the RESTlet is deployed, NetSuite requires the call to include an authorization header. For details about adding an authentication header to a RESTlet, see [RESTlet Authentication](#).

For most RESTlet calls, you must also include a content-type header, which tells NetSuite how your request body will be formatted and how NetSuite should format its response. For details, see [Creating a Content-Type Header](#).

Deploying a RESTlet

Before you can use a RESTlet, you must follow the same guidelines as you would with most entry point scripts. At a high level, you must do the following:

- [Make Sure the Script Is Formatted Properly](#)
- [Create a Script and Script Deployment Record](#)

Make Sure the Script Is Formatted Properly

All of the following must be true:

- The script must have the correct structure for SuiteScript 2.0. For example, the script must have an interface that includes at least one entry point appropriate for the RESTlet script type. The script must also contain a corresponding entry point function. For details on how to structure a SuiteScript 2.0 script, see [Correct Structure for Entry Point Scripts](#). Note that the entry points you use will determine how the RESTlet can be called. For details on the RESTlet entry points, see [RESTlet Script Entry Points](#).
- The script must use the required JSDoc tags. The @NScriptType must be `RESTlet` (or `Restlet`; these values are not case-sensitive). For further details on the required JSDoc tags, see [Required JSDoc Tags for Entry Point Scripts](#).

Create a Script and Script Deployment Record

Before you can use a RESTlet, you must upload it to your File Cabinet. Then you use the file to create a script record and a script deployment record.

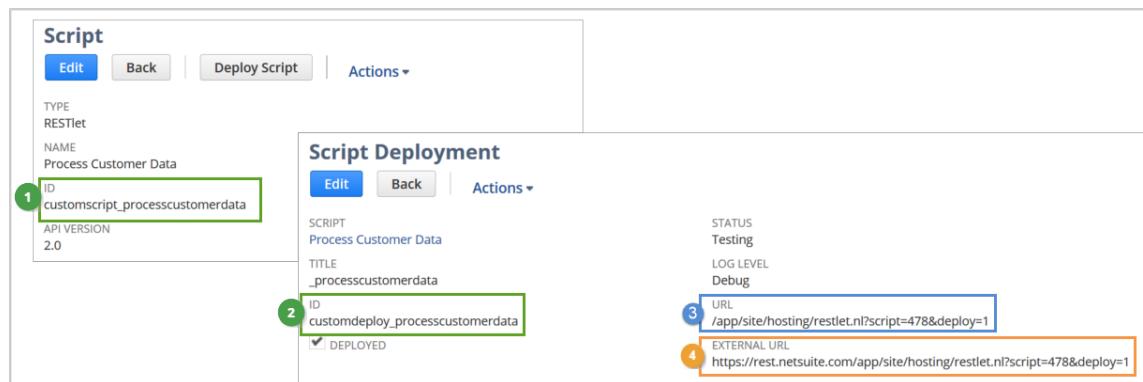
For general help creating script records and script deployment records, see [SuiteScript 2.0 Record-Level Scripts](#).

When creating these records for a RESTlet, be aware of the following:

- NetSuite recommends that you enter meaningful data in the script record's ID field and the script deployment record's ID field. When you save the records, the system creates IDs that include the text you entered. One possible use of these IDs is to identify the RESTlet when calling it from another SuiteScript. For this reason, it may be helpful to have created meaningful ID strings.
- Unlike some other script types, you do not deploy a RESTlet for any particular record type. Each RESTlet is available independently of any particular record type or record instance.
- The script deployment record includes a field called Status, which has possible values of Released and Testing. Before you can call the RESTlet from an external source, the Status field must be set to Released.
- When you save a script deployment record for a RESTlet, the system automatically generates a partial and full URL that you can use to call the RESTlet. However, if you are calling the RESTlet from within an integration and you want to use the full URL, you must include logic that dynamically discovers the RESTlet domain. For more information, see [Identifying a RESTlet in a Call](#).

Identifying a RESTlet in a Call

Before you can access a RESTlet, you must know how to identify it in your call. In general, you use values from the script deployment record. In some cases, you may also have to use the ID value from the script record.



For details, see the following sections:

- [Internal Versus External Calls](#)
- [Dynamically Generating a Full URL](#)

Internal Versus External Calls

The values you use to identify a RESTlet vary depending on the source of the call. For details, see the following table. Each number in the table refers to a callout in the screenshot above.

Source of the Call	Identify the RESTlet with:
An external client	A full URL, similar to the one shown on the script deployment record, in the External URL field (4). For details, see Dynamically Generating a Full URL .
A client SuiteScript with an active session in the same NetSuite account where the RESTlet is deployed	<p>Either of the following:</p> <ul style="list-style-type: none"> ■ The partial URL shown on the script deployment record, in the URL field (3). For an example, see Example: Client Script that Calls a RESTlet. ■ A URL generated by using the N/url module. To generate the URL this way, you need the script ID, which is viewable on the script record in the ID field (1), in combination with the deployment ID, which is viewable on the script deployment record, in its ID field (2). For an example, see Example: Suitelet that Calls a RESTlet.
A server SuiteScript in the same NetSuite account where the RESTlet is deployed	<p>A full URL, similar to the one shown on the script deployment record, in the External URL field (4). This URL must be dynamically generated, in one of the following ways:</p> <ul style="list-style-type: none"> ■ By using the REST roles service. For details, see Using the roles Service to Dynamically Discover Domains. ■ By using the N/url module. To generate the URL this way, you need the script ID, which is viewable on the script record in the ID field (1), in combination with the deployment ID, which is viewable on the script deployment record, in its ID field (2). For an example, see Example: Suitelet that Calls a RESTlet.
A server SuiteScript in a different NetSuite account from where the RESTlet is deployed	A full URL, similar to the one shown on the script deployment record, in the External URL field (4). This URL must be dynamically generated by using the REST roles service. For details, see Dynamically Generating a Full URL .



Note: If you are calling a RESTlet by using the delete or get method, you must extend the URL to include any input data that is required by the RESTlet's logic. For details, see [Selecting an HTTP Method for Calling a RESTlet](#).

Dynamically Generating a Full URL

When you save a script deployment record for a RESTlet, the system automatically generates a full URL that can be used to call the RESTlet. This value is shown in the **External URL** field.

However, in general, you should not hard-code this URL in a script, or in any other part of your integration. Instead, you should create logic that dynamically generates the portion of the URL that represents the RESTlet domain.

The RESTlet domain is the first part of the URL. It appears as https://rest.netsuite.com, https://rest.na1.netsuite.com, or a similar variant. You must dynamically generate this portion of the URL because it can change. The domain can change because NetSuite hosts accounts in multiple data centers, and accounts are sometimes moved. The External URL value on the script deployment record is correct and will be updated if your account moves. Your integration must also be adaptable.

Use the following approaches:

- For calling a RESTlet from an external source, use NetSuite's roles service. For details on using this service, see [The REST roles Service](#).

- If you are calling a RESTlet from within NetSuite, you can use the N/url module. With this approach, you provide the ID values from the script record and script deployment record. For an example, see Example: Suitelet that Calls a RESTlet.

Selecting an HTTP Method for Calling a RESTlet

When you call a RESTlet, you can use one of four supported HTTP methods: delete, get, post, or put.

For more details on using these methods with RESTlets, see the following sections:

- A Call's Method Must Match an Entry Point
- HTTP Method Functionality
- Input Data Is Handled Differently by Different Methods

A Call's Method Must Match an Entry Point

For each supported HTTP method there is a corresponding supported entry point. For a call to be successful, the method used in the call must match an entry point defined in the RESTlet's interface.

The snippets shown in the following screenshot include a successful pairing of a RESTlet and a call to that RESTlet: In the first snippet, the Suitelet calls the RESTlet by using the get method. Because the RESTlet's interface includes a get entry point, as shown in the second snippet, the call would be successful.

```
// Here, the Suitelet calls a RESTlet
var response = https.get ({url: url, headers: headers});

// In this function, the RESTlet retrieves
// a standard NetSuite record.

function _get(context) {
    doValidation([context.recordtype, context.id], ['recordtype', 'id'], 'GET');
    return record.load({
        type: context.recordtype,
        id: context.id
    });
}

return {
    get : _get,
};

}
```

HTTP Method Functionality

The following HTTP methods are supported: delete, get, post, and put. These methods are defined at the following link:

<https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html>

However, be aware that the way a RESTlet behaves is defined in the RESTlet script and may not necessarily correlate with the intended behavior of the HTTP method being used. Although it is not recommended, your RESTlet could for example define an entry point get function that creates data,

rather than retrieving it. Similarly, whether or not a method requires data varies depending on how you write the function. You could write a post entry point function that does not require input.

Input Data Is Handled Differently by Different Methods

The way you pass input parameters to a RESTlet varies depending on the HTTP method you use, as described in the following table.

Method	Placement of required input
delete	Arguments must be embedded in the URL used to make the call.
get	
post	Arguments must be included in a request body written in JSON (JavaScript Object Notation) or plain text.
put	



Note: For examples of requests, see [RESTlet Script and Request Samples](#).

Creating a Content-Type Header

Depending on the design of a RESTlet, calls to that RESTlet may require a request body. Additionally, sometimes data is returned by a RESTlet, and you may want to use that data.

In the first situation, you must tell NetSuite how your request body will be formatted. In the second, you might want to specify the format of the data that is returned. Note that both formats must be the same. You control this choice by adding a content-type header to your request.

For details on the supported values for this header, see the following table.

Value	Notes
application/json	JSON is the appropriate choice for most RESTlets that require a request body, because it lets you map values to fields.
application/xml	XML is supported only for the get method.
text/plain	Because plain text does not let you map values to fields, this choice should be used only for RESTlets that require limited and simple input.

Structuring the Header

Format your content-type header as follows:

```
Content-Type: application/json
```

Note that content-type header values are case-sensitive.

Error Handling

If you omit a content-type header, the request fails with an HTTP error code reading 206: Partial Content.

RESTlet Authentication

NetSuite requires authentication for all RESTlet calls, including calls from each of the following:

- An application outside of NetSuite.
- A server-side script deployed in a different NetSuite account from where the RESTlet is deployed.
- A client script deployed in the same account where the RESTlet is deployed.

In the last case, authentication is automatic, because a client script already has an active session. However, the first two types of calls require that you include an authorization header.

Two types of authorization headers are supported. They are described in the following topics:

- [Using User Credentials for RESTlet Authentication](#)
- [Using TBA for RESTlet Authentication \(OAuth\)](#)

i Note: RESTlets are part of SuiteScript. They are **not** part of NetSuite's web services feature. Be aware that if a role has the Web Services Only option set to true, a user logged in through that role is permitted to send web services calls only. RESTlet calls receive an INVALID_LOGIN_CREDENTIALS error response.

Using User Credentials for RESTlet Authentication

When you call a RESTlet, you can authenticate by providing a user ID and password. With this approach, you use an NLAuth authorization header. NLAuth is a NetSuite-specific authentication approach that is used for RESTlets only.

i Note: If you are calling a RESTlet from an external source, you must authenticate by using either user credentials or token-based authentication. For details on TBA, see [Using TBA for RESTlet Authentication \(OAuth\)](#).

Required Data

To construct an NLAuth authorization header, you use the fields described in the following table.

Field	Description	Required?	Notes
nlauth_account	The ID of the NetSuite account where the RESTlet is deployed	Yes	
nlauth_email	The email address with which the user logs in to NetSuite	Yes	
nlauth_signature	The user's password	Yes	
nlauth_role	The internal ID of a role with which the user is associated	No	If you omit this value, the system selects a role based on the logic described in FAQ: Customer Center Roles Segregation . See the answer for What effect does this change have on login through SuiteTalk (web services) and RESTlets?



Important: RESTlet authentication accepts special characters **only** if they are URL encoded. If your credentials contain special characters, replace each special character with its appropriate URL encoding. For additional information on URL encoding, see http://www.w3schools.com/tags/ref_urlencode.asp.

Syntax

The NLAuth header requires the following elements:

- The prefix `NLAuth`, followed by a space.
- A series of field-value pairs. Each pair should include the field name, an equals sign, and a value. Separate the pairs by commas. NetSuite recommends entering the key-value pairs without leading or trailing spaces. For example, `nlauth_account=123456`, rather than `nlauth_account= 123456 ,`

Example

The following snippet shows a correctly formatted NLAuth header.

```
Authorization: NLAuth nlauth_account=12345, nlauth_email=jsmith@ABC.com, nlauth_signature=xxxx,
nlauth_role=4
```

Using TBA for RESTlet Authentication (OAuth)

If appropriate, you can use NetSuite's Token-Based Authentication feature to authenticate when calling RESTlets. With this approach, you use the OAuth 1.0 specification to construct an authorization header. For details, see the following topics:

- [TBA Setup Requirements](#)
- [Required Data for Using TBA with RESTlets](#)
- [Example OAuth Header](#)
- [Tracking RESTlet Calls Made with TBA](#)



Note: If you are calling a RESTlet from an external source, you must authenticate by using either TBA or user credentials. For details on user credentials, see [Using User Credentials for RESTlet Authentication](#).

TBA Setup Requirements

Using the OAuth protocol with RESTlets requires NetSuite's Token-based Authentication (TBA) feature. Before you can use TBA, you must complete several setup tasks. These tasks include the following:

- You must have enabled the Token-based Authentication feature. For details, see the help topic [Enabling the Token-based Authentication Feature](#).
- You must have created a role that permits logging in using token-based authentication. For details, see the help topic [Setting Up Token-based Authentication Roles](#).
- You must have assigned a user to a role that has permission to log in by using token-based authentication. For details, see the help topic [Assigning Users to Token-based Authentication Roles](#).
- An integration record representing the sending application must exist at **Setup > Integration > Manage Integrations**. On the integration record, the Token-based Authentication option must be

enabled. Enabling this option causes the system to generate the consumer key and secret that represent the application. For details, see [Creating an Integration Record](#).

- You must have the consumer key and secret that were generated when the integration record's Token-based Authentication option was enabled. If you do not have these credentials, you can generate new ones. For details, see [Regenerating a Consumer Key and Secret](#).
- You must have created a token and token secret for the user who will call the RESTlet. For details on this process, see the help topic [Managing TBA Tokens](#).

 **Note:** For general details about NetSuite's Token-based Authentication feature, see the help topic [Token-based Authentication](#).

After you have verified that the prerequisite steps have been completed, you can create logic for generating an OAuth header. For details on the data required for creating the header, see [Required Data for Using TBA with RESTlets](#).

Required Data for Using TBA with RESTlets

An OAuth 1.0 RESTlet authorization header requires the data described in the following table. Some of these values can be obtained from the NetSuite UI. Other values must be calculated. Typically, your integration should include logic to identify these values and generate the finished header. Follow the [OAuth 1.0 protocol](#) to create the authorization header.

Field	Description	Notes
realm	The ID of the NetSuite account where the RESTlet is deployed.	You can find this value at Setup > Company > Company Information, in the Account ID field.
oauth_consumer_key	The consumer key for the integration record being used to track the calling application. This string was created when you checked the Token-based Authentication box on the integration record and saved it. To create an OAuth header, you also need the consumer secret that goes with the key, although you do not use the secret explicitly. If you no longer have these values, you can regenerate them. For details, see Regenerating a Consumer Key and Secret .	
oauth_token	A token that represents a unique combination of a user and an integration record. This string can be generated in multiple ways. For details, see the help topic Managing TBA Tokens . To create an OAuth header, you also need the token secret that goes with the token, although you do not use the secret explicitly.	
oauth_nonce	A unique, randomly generated alphanumeric string of 6-64 characters.	
oauth_timestamp	A current timestamp in Unix format.	
oauth_signature_method	A hash algorithm that can be used to create an RFC 2104-compliant signature.	Supported choices are:

Field	Description	Notes
		<ul style="list-style-type: none"> ■ HMAC-SHA1 ■ HMAC-SHA256
oauth_version	The version of OAuth being used.	Only one value is supported: 1.0.
oauth_signature	A signature generated as described in Section 3.4 of RFC 5849, which describes the OAuth 1.0 specification: https://tools.ietf.org/html/rfc5849 .	<p>To create the signature, you need all of the other values listed in this table, among others, such as the HTTP method being used to make the call.</p> <p>With many languages, an OAuth library is available to help you create the signature. For details about some of the third-party open source libraries that are available, see SuiteAnswer 42171.</p> <p>If you are working in a language that does not have a library, you may want to refer to SuiteAnswer 42019 for an overview of the signature-creation process.</p>

For an example of the finished header, see [Example OAuth Header](#).



Important: To prevent issues due to out of synch time, keep time on your servers synchronized using Network Time Protocol (NTP).

Example OAuth Header

The following snippet shows a correctly formatted OAuth header.

```
Authorization:
OAuth realm="12345",
oauth_consumer_key="4a3ff6c251a55057bb1e62d8dc8998a0366e88f3a8fe735265fc425368b0f154",
oauth_token="52fce88fecf2e2b74e833e7dfc4cae79ff44c3ca9f696d61e2a7eac6c8357c3c",
oauth_nonce="qUwlmpvtGCS4sHJe8F7x",
oauth_timestamp="1462453273",
oauth_signature_method="HMAC-SHA1", oauth_version="1.0",
oauth_signature="8PI9lIYxUmUONjxFJUSMD9o0mc%3D"
```

Tracking RESTlet Calls Made with TBA

If you use OAuth headers when calling RESTlets, you have the ability to track and block RESTlet calls. You manage RESTlet activity by using integration records. Each record shows the RESTlet calls that authenticated by using that record's consumer key.

Integration records are located at Setup > Integration > Manage Integrations. For more information on using integration records in conjunction with RESTlets, see the following topics:

- Creating an Integration Record
- Blocking an Application
- Enabling an Application to Use Token-based Authentication
- Regenerating a Consumer Key and Secret
- Using the RESTlets Execution Log
- Ownership of Integration Records
- Tracking Changes to Integration Records

i Note: For more information about managing integration records, see the help topic [Managing Integrations](#), which is part of the SuiteTalk (Web Services) Platform Guide. However, be aware that some of the detail in that guide pertain only or primarily to web services.

Creating an Integration Record

To create an OAuth header, you must have a consumer key and secret that represents the application that will call the RESTlet. In general, you create these values by creating an integration record. When you create the record and enable the record's Token-based Authentication option, the system generates and displays the consumer key and secret. These values are shown only one time. However, if you lose the values, you can regenerate them, as described in [Regenerating a Consumer Key and Secret](#).

In some cases, you might be working with a partner that has provided an integration record to you through a bundle. In this situation, you obtain the consumer key and secret from the partner. If you have an integration record that was installed through a bundle, note that not all fields on the record are modifiable. Non-modifiable fields include Name, Description, User Credentials, and Token-based Authentication.

To create an integration record, use the following procedure.

i Note: The integration record is also used to track web services activity. Not all fields on the record are relevant to RESTlet activity. For full details on how this record is used with web services, see the help topic [Managing Integrations](#).

To manually create an integration record:

1. Go to Setup > Integration > Manage Integrations > New.
2. In the **Name** field, enter a name for the application.
3. If appropriate, enter a description in the **Description** field.
4. If you want NetSuite to block requests from this application, set the **State** dropdown list to **Blocked**. Otherwise, leave this field set to its default of **Enabled**. If you intend to distribute this record, be aware that the value you choose for this field is not propagated to accounts where the record is installed. The value of this field is always specific to one NetSuite account. If the record is installed by bundling, the State field is always initially set to Enabled.
5. If appropriate, enter additional details about the application in the **Note** field. If you intend to distribute this record, be aware that the text you enter in this field is not visible in accounts where the record is installed. The value of this field is always specific to one NetSuite account.
6. On the **Authentication** subtab, select the authentication methods that this application should be permitted to use. You can choose one or both of the following:
 - **User credentials** – This option is applicable only to web services activity. For help using this option in conjunction with web services, see the help topic [Sending an Application ID with User Credentials](#).

- **Token-based authentication** – This option lets the application use token-based authentication (TBA).
7. Click **Save**.

The system saves the new record. If you selected the **Token-based Authentication** option, the updated page also shows the record's consumer key and secret.

 **Warning:** For security reasons, the only time the Consumer Key and Consumer Secret values are displayed is on the confirmation page. After you leave this page, these values cannot be retrieved from the system. If you lose or forget these credentials, you must reset them to obtain new values. Treat these values as you would a password. Never share these credentials with unauthorized individuals and never send them by email.

Blocking an Application

If appropriate, you can block an application represented by an integration record. Blocking an application has the following effects:

- The application is blocked from authenticating using the consumer key associated with the integration record. So, if an application is using an OAuth header to call RESTlets (using data from this integration record), these calls will be blocked,
- If the application makes web services requests, the requests are blocked if they reference either the consumer key or the application ID associated with the integration record.

Note that this procedure does not prevent an application from calling a RESTlet by using the NLAuth authentication method. Similarly, the application is not blocked if it already has an existing session.

To block an application:

1. Navigate to Setup > Integration > Managing Integrations, and open the appropriate integration record for editing.
2. Set the **State** field to **Blocked**.
3. Click **Save**.

Enabling an Application to Use Token-based Authentication

In some cases, you might have an existing application that is not set up for token-based authentication. For example, an integration record might have been created to track web services activity, and that application might authenticate through user credentials. If appropriate, you can enable token-based authentication for that application.

 **Note:** If the integration record was created in another account and installed in your account through a bundle, you cannot modify the Token-based Authentication field. For more details, see [Ownership of Integration Records](#).

To enable token-based authentication for an existing application:

1. Navigate to Setup > Integration > Managing Integrations, and open the appropriate integration record for editing.
2. Check the **Token-based Authentication** box.
3. Click **Save**.

The system displays the consumer key and secret on the screen. Make a note of these values. You will need them to create an OAuth header.

 **Warning:** For security reasons, the only time the Consumer Key and Consumer Secret values are displayed is on the confirmation page. After you leave this page, these values cannot be retrieved from the system. If you lose or forget these credentials, you must reset them to obtain new values. Treat these values as you would a password. Never share these credentials with unauthorized individuals and never send them by email.

Regenerating a Consumer Key and Secret

If you no longer have the consumer key and secret for an integration record that was created in your NetSuite account, you can regenerate them. Be aware that when you reset these credentials, the old ones are invalidated.

At the same time, even after you reset the consumer key and secret, you can still use a token and token secret that were created with the original consumer data. But they must be used with the new consumer key and secret.

 **Note:** If the integration record was created in another account and installed in your account through a bundle, you cannot reset the credentials. The credentials can be reset only by an authorized user in the NetSuite account where the record was created. For more details, see Ownership of Integration Records.

To regenerate a consumer key and secret:

1. Go to Setup > Integration > Manage Integrations.
2. Select the record for which you want to generate a new consumer key and secret.
The record opens in view mode.
3. Click the **Edit** button.
4. Click the **Reset Credentials** button.
The system displays a popup message asking if you are sure you want to reset the credentials.
5. Click **OK**.
The system resets the credentials. The record is again shown in view mode, with the new consumer key and secret displayed.

 **Warning:** For security reasons, the only time the consumer key and consumer secret values are displayed is on the confirmation page. After you leave this page, these values cannot be retrieved from the system. If you lose or forget these credentials, you must reset them to obtain new values. Treat these values as you would a password. Never share these credentials with unauthorized individuals, and never send them by email.

Using the RESTlets Execution Log

Each integration record includes a subtab labeled RESTlets Execution Log. This log lists RESTlet calls that are uniquely identified with that integration record. That is, the log includes those requests that use token-based authentication and reference the integration record's consumer key.

 **Note:** Calls made using the NLAuth method of authentication are not logged on any integration record.

For each logged request, the RESTlets Execution Log includes details such as the following:

- The date and time that the call was made.
- The duration of the request.
- The email address of the user who made the request.

- The action taken.
- The corresponding script ID and deployment ID.

Ownership of Integration Records

When you create an integration record, it is automatically available to you in your NetSuite account. Your NetSuite account is considered to be the owner of the integration record, and the record is fully modifiable by administrators in your account.

You can also install records in your account that were created elsewhere. For example, an integration record could be bundled and distributed. If you install a bundle that includes an integration record, the record is considered to be an installed record. It is owned by a different NetSuite account. On such records, you can make changes to only two fields: the Note field and the State field. All other fields, including the authentication and Description fields, can be changed only by an authorized user in the account that owns the record. When the owner makes changes to these fields, the new settings are pushed automatically to your account. These changes are not reflected in the system notes that appear in your account.

Tracking Changes to Integration Records

If you want to review changes that were made to an integration record, you can refer to the system notes for that record. System notes are used to track events such as the creation of the record, the initial values of its fields, and subsequent updates. For example, if a user changed the State field from Blocked to Enabled, a system note would provide a record of that change.

For each event, the system records details such as the ID of the user who made the change and the timestamp of the change. If a user assigns a value to a field that already had a value, the system note also shows the field's former setting.

Be aware that in general, system notes are created only for those fields that you are permitted to change. For additional details, see the help topic [Special Cases Related to System Notes Logging](#). Note that some of the information in that topic is specific to web services.

You can locate system notes for integration records in either of the following ways:

- By using the System Note search type, at Reports > New Search.



- By clicking on the System Notes subtab of any integration record.

System Notes						
FIELD	VIEW	- All -	Default			
Customize View						
DATE	SET BY	CONTEXT	TYPE	FIELD	OLD VALUE	NEW VALUE
12/28/2015 1:48 pm	Kate Johnson	UI	Change	State	Enabled	Blocked
12/28/2015 1:48 pm	Kate Johnson	UI	Change	Last State Change By	Jack Smith	Kate Johnson
12/28/2015 12:38 pm	Susan Jones	UI	Set	Note	See Susan for issues related to this integration.	
12/28/2015 12:38 pm	Susan Jones	UI	Set	Description	This integration is used to manage sales orders and customers.	
12/28/2015 12:33 pm	Jack Smith	UI	Set	Token-Based Authentication	F	

RESTlet Reference

See the following topics for more details about working with RESTlets:

- [RESTlet Error Handling](#)
- [RESTlet Governance](#)
- [RESTlet Security](#)
- [The REST roles Service](#)

RESTlet Error Handling

For details about error handling for RESTlets, refer to the following sections:

- [Supported HTTP Status Codes for RESTlets](#)
- [SuiteScript Errors Returned by RESTlets](#)

Supported HTTP Status Codes for RESTlets

For details about the HTTP status codes supported for RESTlets, see:

- [HTTP Success Code](#)
- [HTTP Error Codes](#)

HTTP Success Code

NetSuite supports one HTTP success code for RESTlets: **200 OK**. This code indicates that the request was executed successfully. Note that this code does not necessarily mean that your request worked as you intended. In some cases, a SuiteScript error might occur and be successfully handled by the RESTlet script. In these cases, an HTTP code of 200 is used, and details of the error are described in the response body.

HTTP Error Codes

NetSuite supports the following HTTP error codes for RESTlets.

Code	Description
206 Partial Content	You receive this error if you attempt to use a content-type that is unsupported for the HTTP method you are using.
302 Moved Temporarily	This error can be displayed if your request went to the wrong NetSuite data center. Review your integration and make sure that it generates the NetSuite RESTlet domain dynamically. For details, see Dynamically Generating a Full URL .
400 Bad Request	The RESTlet request failed with a user error. Any errors encountered at run time that are unhandled return a 400 error. (If the user code catches the error, a status code of 200 is returned.)
401 Unauthorized	There is no valid NetSuite login session for the RESTlet call.
403 Forbidden	The RESTlet request was sent to an invalid domain, meaning a domain other than https://rest.netsuite.com , https://rest-na1.netsuite.com , or a similar variant.
404 Not Found	A RESTlet script is not defined in the RESTlet request.

Code	Description
405 Method Not Allowed	The request method used is not valid.
415 Unsupported Media Type	An unsupported content type was specified.
500 Internal Server Error	This error occurs for non-user errors that cannot be recovered by resubmitting the same request. If this type of error occurs, contact Customer Support to file a case.
503 Service Unavailable	The NetSuite database is offline, or a database connection is not available.

SuiteScript Errors Returned by RESTlets

In some cases, the response to a RESTlet is a SuiteScript error. For details, see the following sections:

- [SuiteScript Error Codes Used by RESTlets](#)
- [SuiteScript Error Message Formatting for RESTlets](#)

SuiteScript Error Codes Used by RESTlets

The following table describes some of the supported SuiteScript errors that can occur when using RESTlets.

Code	Description	Notes
INVALID_LOGIN_ATTEMPT	Invalid login attempt.	This error indicates a problem in an OAuth header. It can be returned when the nonce, consumer key, token, or signature in the OAuth header is invalid.
INVALID_LOGIN_CREDENTIALS	You have entered an invalid email address or account number. Please try again.	This error indicates a problem in an NLAUTH header.
INVALID_REQUEST	The request could not be understood by the server due to malformed syntax.	This error is returned because of malformed syntax in an OAuth header. For example, this error can occur when the signature method, version, or timestamp parameter is rejected.
INVALID_RETURN_DATA_FORMAT	You should return {1}.	This error is used if the response data does not match the expected format, as specified by the content-type header.
SSS_INVALID_SCRIPTLET_ID	That Suitelet is invalid, disabled, or no longer exists.	If you receive this error, make sure that the URL points to the correct RESTlet script deployment ID.
UNEXPECTED_ERROR	An unexpected error occurred. Error ID: {1}	

SuiteScript Error Message Formatting for RESTlets

The following examples illustrate SuiteScript errors formatting for each supported content-type.

JSON

```
{
  "error": {
    "code": "SSS_INVALID_SCRIPTLET_ID",
    "message": "That Suitelet is invalid, disabled, or no longer exists."
  }
}
```

XML

```
<error>
  <code>SSS_INVALID_SCRIPTLET_ID</code>
  <message>That Suitelet is invalid, disabled, or no longer exists.</message>
</error>
```

Text

```
error code: SSS_INVALID_SCRIPTLET_ID
error message: That Suitelet is invalid, disabled, or no longer exists.
```

RESTlet Governance

The SuiteScript governance model tracks usage units on two levels: API level and script level. At the API level, RESTlets have the same usage limits as other types of SuiteScripts. At the script level, RESTlets allow 5,000 usage units per script, a limit five times greater than Suitelets and most other types of SuiteScripts. For more information, see the help topic [SuiteScript Governance](#).

There is a limit of 10MB per string used as RESTlet input or output.

SuiteScript currently does not support a logout operation similar to the one used to terminate a session in SuiteTalk.

RESTlet Security

The URLs for accessing RESTlets are protected by TLS encryption. NetSuite supports TLS 1.0, 1.1, and 1.2 encryption for all RESTlets (and for all NetSuite domains). Only requests sent using TLS encryption are granted access.

The REST roles Service

NetSuite provides a REST service called roles, which you can use to retrieve the following information:

- A list of roles that a user belongs to.
- The correct domains for external client access to a NetSuite account. In an integration, domains must be discovered dynamically, because they can change without notice.

You call the roles service by sending a request to one of the roles service URLs, as described in [URLs for Accessing the REST roles Service](#). The request must use the get method, and it must also include an NLAuth authorization header. For details, see [Authentication for the REST roles Service](#). If you need to

retrieve Customer Center roles, note that you must include a NetSuite account ID in the authorization header.

In response, the service returns data about roles and domains. For examples, see [Sample Responses from the roles Service](#).

For specific details on domain data returned by the roles service, see [Domain Data Returned by the roles Service](#).

Authentication for the REST roles Service

Each call to the REST roles service must include an NLAuth authorization header. This header must identify a user and the user's password. Additionally, if you need to retrieve data about Customer Center roles, the account ID **must** be included in the header. Otherwise, the account ID is optional. Note the following:

- If your request includes a NetSuite account ID, the information returned is specific to that account. All roles are returned for the user, including the Customer Center roles, and any roles created based on the Customer Center role.
- If your request omits a NetSuite account ID, the system returns details for every NetSuite account to which the user identified in the header has access. All roles except Customer Center roles are included.

For examples, see [Sample Responses from the roles Service](#).

For details on domain data returned by the roles service, see [Domain Data Returned by the roles Service](#).

For information about constructing an NLAuth authorization header, see [Using User Credentials for RESTlet Authentication](#).

URLs for Accessing the REST roles Service

To use the roles service, you send a get request to the appropriate URL. However, the URL that you use can vary in the following ways:

- You use a different URL for each of the following: production, sandbox, and release-preview environments. For details, see [Sample URLs](#).
- In some cases, your request may be redirected. If you have an integration that calls the roles service, your integration must include logic to handle this redirection. For details, see [Calls that May Require Redirection](#).

Sample URLs

You can use the URLs described in the following table to call the roles service.

Environment	URL
Production and EU Sandbox (sandboxes hosted in European Union data centers)	https://rest.netsuite.com/rest/roles and similar variants
Release Preview	https://rest.beta.netsuite.com/rest/roles
EU Release Preview (Release Preview for accounts hosted in North American data centers)	https://rest.eu1.netsuite.com/rest/roles

Environment	URL
NA Sandbox (sandboxes hosted in North American data centers)	https://rest.sandbox.netsuite.com/rest/roles



Note: NetSuite maintains multiple URLs for the production version of the roles service. Each data center hosts the service on the REST domain specific to that data center. However, you are not expected to know the data center of your account when you send the request. For that reason, you can use any production URL. But be aware that, in some cases, your request may be directed to a different URL, as described in the next section. For a list of all REST service domains, see the help topic [Understanding NetSuite URLs and Data Centers](#).

Calls that May Require Redirection

Sometimes calls to the roles service include a NetSuite account ID in the authorization header. If you send this type of request to a production version of the roles service, the call may be redirected to a different URL. Redirection can occur because these requests must be handled by the same data center that hosts the NetSuite account. For this reason, if you have an integration that makes this type of call, the integration must include logic to handle redirection.

For example, suppose you are calling the service to retrieve roles information for a user in your NetSuite account. You may not know which data center your account is hosted in, especially if you have never called the roles service before. This knowledge gap is expected. The service is designed so that you can send your request to any of the available production URLs for the service. However, if your account is hosted in the EU and you send the request to a URL associated with a North American data center, your request will be redirected to <https://rest.eu2.netsuite.com>. For this reason, your integration must include logic for handling the 302 Found response status code, which is the code used when redirection occurs.

By contract, if your authorization header omits a NetSuite account ID, your request is handled without redirection.

Domain Data Returned by the roles Service

The REST roles service lets you dynamically discover the following types of domains for any NetSuite account.

Type of Domain	Examples
RESTlet	https://rest.netsuite.com , https://rest.na1.netsuite.com , and similar variants
System	https://system.netsuite.com , https://system.na1.netsuite.com , and similar variants
Web services	https://webservices.netsuite.com , https://webservices.na1.netsuite.com , and similar variants

It is important to dynamically discover these domains because, for any NetSuite account, these domains can change. They can change because NetSuite hosts data in multiple data centers, and the location of your account data can change. Therefore, any integration that includes full URLs **must** include logic for dynamically discovering these domains. A hard-coded URL could fail.

Sample Responses from the roles Service

The following sections show sample responses from the REST roles service:

- Calls that Include a NetSuite Account ID
- Calls that Omit a NetSuite Account ID

Calls that Include a NetSuite Account ID

A typical call to the roles service includes a NetSuite account ID in its authorization header. For example, the header might look like the following:

```
NLAUTH nlauth_account=023456, nlauth_email=john@smith.com, nlauth_signature=Welcome123
```

In response, the system returns data specific to that NetSuite account and user, as shown in the following example.

```
[
{
  "account": {
    "internalId": "023456"
    "name": "Account 1"
  }
  "role": {
    "internalId": 14
    "name": "Customer Center"
  }
  "dataCenterURLs": {
    "webservicesDomain": "https://webservices.na2.netsuite.com"
    "restDomain": "https://rest.na2.netsuite.com"
    "systemDomain": "https://system.na2.netsuite.com"
  }
}
]
```

With this approach, the system returns **all** roles for the user in the specified account. Results include the Customer Center role and any custom roles created based on the Customer Center role, if the user belongs to those roles. If the account ID is omitted, the service does not return Customer Center roles.

Calls that Omit a NetSuite Account ID

A call to the roles service can omit a NetSuite account ID. With these calls, the authorization header includes only user and password data, as follows:

```
NLAUTH nlauth_email=john@smith.com, nlauth_signature=Welcome123
```

With this type of call, the system returns data on all accounts to which the user has access. However, the system does not return the Customer Center role, nor any custom role that was created based on the Customer Center role. To have the system return Customer Center roles, use the method described in [Calls that Include a NetSuite Account ID](#).

The following snippet shows a sample response to a request that omitted a NetSuite account ID.

```
[
{
  "account": {
    "internalId": "023456"
    "name": "Account 1"
  }
}
```

```

    }
    "role": {
      "internalId": 3
      "name": "Administrator"
    }
  "dataCenterURLs": {
    "webservicesDomain": "https://webservices.na2.netsuite.com"
    "restDomain": "https://rest.na2.netsuite.com"
    "systemDomain": "https://system.na2.netsuite.com"
  }
},
{
  "account": {
    "internalId": "123456"
    "name": "Account 2"
  }
  "role": {
    "internalId": 1
    "name": "Accountant"
  }
  "dataCenterURLs": {
    "webservicesDomain": "https://webservices.netsuite.com"
    "restDomain": "https://rest.netsuite.com"
    "systemDomain": "https://system.netsuite.com"
  }
},
{
  "account": {
    "internalId": "123456"
    "name": "Account 2"
  }
  "role": {
    "internalId": 3
    "name": "Administrator"
  }
  "dataCenterURLs": {
    "webservicesDomain": "https://webservices.netsuite.com"
    "restDomain": "https://rest.netsuite.com"
    "systemDomain": "https://system.netsuite.com"
  }
}
]

```

RESTlet Script and Request Samples

For sample RESTlet scripts and requests, see the following sections:

- Example: Hello World
- Example: RESTlet that Can Retrieve, Delete or Create
- Example: RESTlet that Adds Multiple Records

- Example: RESTlet that Manipulates Scheduled Script
- Example: Client Script that Calls a RESTlet
- Example: Suitelet that Calls a RESTlet

Example: Hello World

Using RESTlets can be more complicated than using other script types because, whereas other script types that can be deployed and then will execute as needed, a RESTlet must be deployed and then called. To call a RESTlet successfully, you must correctly identify your NetSuite account's RESTlet domain, use an HTTP method that matches an entry point in your script, and use two required headers: Content-Type and Authorization. (For details on these headers, see [Creating a Content-Type Header](#) and [RESTlet Authentication](#).)

When getting started, you may want to test your RESTlet setup with a simple script such as the following.

RESTlet

```
/** 
 * @NApiVersion 2.x
 * @NScriptType restlet
 */
define([], function() {
  return {
    get : function() {
      return "Hello World!"
    }
  }
});
```

Sample Get Call and Response

You call this RESTlet by using the get method. Because the RESTlet takes no arguments, you would not need to extend the URL with additional values. For testing purposes, you could use the value that appears in the External URL field of the script deployment record. (However, in an integration, remember that you must dynamically discover the RESTlet domain.)

For example, you could call this RESTlet by using a URL like the following — one that does not include embedded parameters:

```
https://rest.netsuite.com/app/site/hosting/restlet.nl?script=482&deploy=1
```

After you add the two required headers, the generated call would look like the following.

```
GET /app/site/hosting/restlet.nl?script=482&deploy=1 HTTP/1.1
HOST: rest.netsuite.com
authorization: NLAuth nlauth_account=12345, nlauth_email=john@smith.com, nlauth_signature=Welcome123
content-type: application/json
cookie: ...
```

The RESTlet would return the following response:

```
Hello World!
```

Example: RESTlet that Can Retrieve, Delete or Create

The following RESTlet includes logic for all supported entry points: get, delete, post, and put.

RESTlet

```
/*
 *@NApiVersion 2.x
 *@NScriptType Restlet
 */
define(['N/record', 'N/error'],
    function(record, error) {
        function doValidation(args, argNames, methodName) {
            for (var i = 0; i < args.length; i++)
                if (!args[i] && args[i] !== 0)
                    throw error.create({
                        name: 'MISSING_REQ_ARG',
                        message: 'Missing a required argument: [' + argNames[i] + '] for method
: ' + methodName
                    });
        }
        // Get a standard NetSuite record
        function _get(context) {
            doValidation([context.recordtype, context.id], ['recordtype', 'id'], 'GET');
            return JSON.stringify(record.load({
                type: context.recordtype,
                id: context.id
            }));
        }
        // Delete a standard NetSuite record
        function _delete(context) {
            doValidation([context.recordtype, context.id], ['recordtype', 'id'], 'DELETE');
            record.delete({
                type: context.recordtype,
                id: context.id
            });
            return String(context.id);
        }
        // Create a NetSuite record from request params
        function post(context) {
            doValidation([context.recordtype], ['recordtype'], 'POST');
            var rec = record.create({
                type: context.recordtype
            });
            for (var fldName in context)
                if (context.hasOwnProperty(fldName))
                    if (fldName !== 'recordtype')
                        rec.setValue(fldName, context[fldName]);
            var recordId = rec.save();
            return String(recordId);
        }
    }
)
```

```
// Upsert a NetSuite record from request param
function put(context) {
    doValidation([context.recordtype, context.id], ['recordtype', 'id'], 'PUT');
    var rec = record.load({
        type: context.recordtype,
        id: context.id
    });
    for (var fldName in context)
        if (context.hasOwnProperty(fldName))
            if (fldName !== 'recordtype' && fldName !== 'id')
                rec.setValue(fldName, context[fldName]);
    rec.save();
    return JSON.stringify(rec);
}
return {
    get: _get,
    delete: _delete,
    post: post,
    put: put
};
});
```

Sample Get Call and Response

To retrieve a record by using this RESTlet, you would use the get method. To identify the record you want to retrieve, you would add values to the URL you use to call the RESTlet. These values would identify the record type and internal ID of the record instance you want. These parameters are defined in the RESTlet's get function as:

- recordtype
- id

You add a value for each parameter by using an ampersand, the name of the parameter, an equals sign, and the parameter's value, as follows:

```
&[name of parameter]=[value]
```

For example, to retrieve a phone call record with the internal ID of 9363, you would use URL like the following:

```
https://rest.netsuite.com/app/site/hosting/restlet.nl?script=474&deploy=1&recordtype=phonecall&id=9363
```

Using the get method in conjunction with this URL would produce the following request:

```
GET /app/site/hosting/restlet.nl?script=474&deploy=1&recordtype=phonecall&id=9363 HTTP/1.1
HOST: rest.netsuite.com
authorization: NLAuth nlauth_account=12345, nlauth_email=john@smith.com, nlauth_signature=Welcome123
content-type: application/json
cookie: ...
```

In response, the system would return data like the following:

```
{
  "id": "9363",
  "type": "phonecall",
  "isDynamic": false,
  "fields": {
    "wfinstances": "",
    "nlloc": "0",
    "nlsub": "1",
    "createddate": "5/18/2016 1:01 am",
    "timezone": "America/Los_Angeles",
    "accesslevel": "F",
    "_eml_nkey_": "143659438",
    "starttime": "",
    "startdate": "5/18/2016",
    "title": "Project kickoff",
    "type": "call",
    ...
  },
  "sublists": {
    "contact": {
      "currentline": {
        "company": "",
        "contact": ""
      }
    }
  }
}
```

Sample Post Call and Response

To use this RESTlet to add a record, you would use the post method. Your arguments must be included in a request body, and the request body would have to be written in JSON rather than plain text. For example:

```
{"recordtype": "phonecall", "type": "phonecall", "title": "Project Kickoff"}
```

You would send your post method to a URL that has not been extended. For testing purposes, you could use the value that appears in the External URL field of the script deployment record. (However, in an integration, remember that you must dynamically discover the RESTlet domain.)

For example, you could call this RESTlet by using a URL like the following — one that does not include embedded parameters:

```
https://rest.netsuite.com/app/site/hosting/restlet.nl?script=474&deploy=1
```

Making a post call using the request body above, plus the appropriate headers, would produce the following request:

```
POST /app/site/hosting/restlet.nl?script=474&deploy=1 HTTP/1.1
HOST: rest.netsuite.com
authorization: NLAuth nlauth_account=12345, nlauth_email=john@smith.com, nlauth_signature=Welcome123
```

```
content-type: application/json
cookie: ...
```

In response, the system returns the internal ID of the newly created record. For example:

```
9564
```

Example: RESTlet that Adds Multiple Records

The following RESTlet example can create several contact records in one call.

RESTlet

```
/**
 * @NApiVersion 2.x
 * @NScriptType restlet
 */
define([ 'N/record' ], function(record) {
    return {
        post : function(restletBody) {
            var restletData = restletBody.data;
            for ( var contact in restletData) {
                var objRecord = record.create({
                    type : record.Type.CONTACT,
                    isDynamic : true
                });
                var contactData = restletData[contact];
                for ( var key in contactData) {
                    if (contactData.hasOwnProperty(key)) {
                        objRecord.setValue({
                            fieldId : key,
                            value : contactData[key]
                        });
                    }
                }
                var recordId = objRecord.save({
                    enableSourcing : false,
                    ignoreMandatoryFields : false
                });
                log.debug(recordId);
            }
        }
    });
});
```

Sample Post Call

To create records with this RESTlet, you would use the post method. You would use a request body to send your values for the new contact records.

```
{"data": [{"firstname": "John", "middlename": "Robert", "lastname": "Smith", "subsidiary": "1"}, {"firstname": "Anne", "middlename": "Doe", "lastname": "Smith", "subsidiary": "1"}]}
```

Example: RESTlet that Manipulates Scheduled Script

The following RESTlet passes a value to a scheduled script.

Note: ThescriptId and deploymentId in this sample are placeholders. Before using this script, replace the IDs with valid values from your NetSuite account.

```
/**  
 * @NApiVersion 2.x  
 * @NScriptType restlet  
 */  
define(['N/task'], function(task) {  
    return {  
        get : function() {  
            var mrTask = task.create({  
                taskType : task.TaskType.SCHEDULED_SCRIPT  
            });  
            mrTask.scriptId = 488;  
            mrTask.deploymentId = 'customdeploy_scheduledscript';  
            mrTask.params = {  
                custscriptcustom_data : 'data'  
            };  
            mrTask.submit();  
            return "Scheduled script updated";  
        }  
    }  
});
```

Example: Client Script that Calls a RESTlet

The following example shows how a client script can call a RESTlet. Because the client script is expected to have an active session, it uses a partial URL to call the RESTlet. That is, the URL does not have to include the RESTlet domain. You can find this partial URL in the **URL** field of the script deployment record for the RESTlet.

Note: The partial URL in this sample is a placeholder. Before using this script, replace this string with a valid value from your NetSuite account.

```
/**  
 * @NApiVersion 2.x  
 * @NScriptType ClientScript  
 */  
define(['N/https'], function(https) {  
    return {  
        pageInit : function() {  
            var dataFromRestlet = https  
                .get('/app/site/hosting/restlet.nl?script=200&deploy=1');  
            console.log(dataFromRestlet.body);  
        }  
    }  
});
```

Example: Suitelet that Calls a RESTlet

The following example shows how a Suitelet can call a RESTlet. Because the Suitelet is deployed in the same NetSuite account as the RESTlet, the script can use the N/url module to resolve the URL for the RESTlet. With this approach, you need the script ID and the script deployment ID associated with the RESTlet.



Note: The scriptId, deploymentId, and authorization data in this sample are placeholders. Before using this script, replace these values with valid data from your NetSuite account.

```
/**  
 * @NApiVersion 2.x  
 * @NScriptType Suitelet  
 */  
define(['N/https', 'N/url'], function(https, urlMod) {  
    return {  
        onRequest : function(options) {  
            var url = urlMod.resolveScript({  
               scriptId: 'customscript196',  
                deploymentId: 'customdeploy1',  
                returnExternalUrl: true,  
                params: {parametername: 'value'}  
            });  
  
            var headers = {'Authorization': 'NLAuth nlauth_account=12345, nlauth_email=john@smith.com, nlauth_signature=Welcome123, nlauth_role=3'};  
  
            var response = https.get({url: url, headers: headers});  
  
            options.response.write(response.body);  
        }  
    };  
});
```

RESTlet Script Entry Points

Script Entry Point	
get	All RESTlet entry points return the HTTP response body.
delete	
put	
post	

delete

Description	Returns the HTTP response body. ■ Returns a string when request Content-Type is 'text/plain'.
-------------	--

	<ul style="list-style-type: none"> ■ Returns an Object when request Content-Type is 'application/json'.
Returns	string object
Since	Version 2015 Release 2

Parameters

i Note: The `requestParams` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Required / Optional	Description	Since
<code>requestParams</code>	Object	required	The parameters from the HTTP request URL. For all content types, parameters are passed into the function as a JavaScript Object.	Version 2015 Release 2

get

Description	<p>Returns the HTTP response body.</p> <ul style="list-style-type: none"> ■ Returns a string when request Content-Type is 'text/plain'. ■ Returns an Object when request Content-Type is 'application/json' or 'application/xml'.
Returns	string object
Since	Version 2015 Release 2

Parameters

i Note: The `requestParams` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Description	Since
<code>requestParams</code>	Object	The parameters from the HTTP request URL. For all content types, parameters are passed into the function as a JavaScript Object.	Version 2015 Release 2

post

Description	<p>Returns the HTTP response body.</p> <ul style="list-style-type: none"> ■ Returns a string when request Content-Type is 'text/plain'.
-------------	--

	<ul style="list-style-type: none"> ■ Returns an Object when request Content-Type is 'application/json'.
Returns	string object
Since	Version 2015 Release 2

Parameters

i Note: The `requestBody` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Required / Optional	Description	Since
<code>requestBody</code>	string Object	required	<p>The HTTP request body.</p> <ul style="list-style-type: none"> ■ Pass the request body as a string when the request Content-Type is 'text/plain' ■ Pass the request body as a JavaScript Object when the request Content-Type is 'application/json'. 	Version 2015 Release 2

put

Description	<p>Returns the HTTP response body.</p> <ul style="list-style-type: none"> ■ Returns a string when request Content-Type is 'text/plain'. ■ Returns an Object when request Content-Type is 'application/json'.
Returns	string object
Since	Version 2015 Release 2

Parameters

i Note: The `requestBody` parameter is a JavaScript object. It is automatically passed to the script entry point by NetSuite.

Parameter	Type	Required / Optional	Description	Since
<code>requestBody</code>	string Object	required	<p>The HTTP request body.</p> <ul style="list-style-type: none"> ■ Pass the request body as a string when the request Content-Type is 'text/plain' ■ Pass the request body as a JavaScript Object when 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			the request Content-Type is 'application/json'.	

Scheduled Script Type

- Scheduled Script Entry Points
- Scheduled Script API
- Scheduled Script Sample
- Using Multiple Queues
- Scheduled Script Governance
- Scheduled Script Execution
- Scheduled Script Deployment
- Scheduled Script Status and Monitoring
- Best Practice: Handling Server Restarts in Scheduled Scripts (SuiteScript 2.0)

Scheduled scripts are server-side scripts that you can deploy to the NetSuite scheduling queue on an ad-hoc or recurring basis.

This script type is suitable for data reporting and maintenance applications. For example, to fetch information on a daily or hourly basis. However, avoid using a scheduled script for long running tasks and batch jobs. Instead, NetSuite recommends that you use a map/reduce script to handle large data processing.

You can also use a scheduled script to work with and then purge temporary records.

Using Multiple Queues

All companies that run NetSuite are provided a **single** queue for running their scheduled scripts. In accounts that have purchased a SuiteCloud Plus license, five queues are available for import jobs, instead of a single queue.

For a summary of SuiteCloud Plus capabilities, see the help topic [SuiteCloud Plus Settings](#).

To learn more about SuiteCloud Plus, or to purchase this add-on module, contact your NetSuite account manager.

Scheduled Script Governance

Scheduled scripts are given a higher limit of usage governance (10,000 units, as opposed to 1,000 units for most other script types).

Scheduled Script Entry Points

Script Entry Point	
execute	Definition of the scheduled script trigger point

For more information about script entry points and SuiteScript 2.0, see [SuiteScript 2.0 Script Types and Entry Points](#).

Scheduled Script API

API	
<code>context.InvocationType</code>	Enumeration that holds the string values for scheduled script execution contexts.

Scheduled Script Sample

This sample shows a script that finds and fulfills sales orders created today.

This sample accepts the search id from a script parameter that it assumes was created with the script record. Prior to running this script, you should create a Sales Order type of saved search. You can use `search.create(options)` and the enum to set up a saved search with the correct search id and search type. You must also set up the script parameter.

Script Sample Prerequisites

1. Create a sales order type of saved search.
2. From the Parameters tab on the script record of the scheduled script, create a parameter. Set the id to "custscript_searchid". Ensure that Type is set to Free-Form Text.
3. From the deployment record, from the Parameters tab, assign the saved search id to the parameter.

```
/**
 *@NApiVersion 2.x
 *@NScriptType ScheduledScript
 */
define(['N/search', 'N/record', 'N/email', 'N/runtime'],
    function(search, record, email, runtime) {
        function execute(context) {
            if (context.type !== context.InvocationType.ON_DEMAND)
                return;
            var searchId = runtime.getCurrentScript().getParameter("custscript_searchid");
            try {
                search.load({
                    id: searchId
                }).run().each(function(result) {
                    log.debug({
                        details: 'transforming so :' + result.id + ' to item fulfillment'
                    });
                    var fulfillmentRecord = record.transform({
                        fromType: record.Type.SALES_ORDER,
                        fromId: result.id,
                        toType: record.Type.ITEM_FULFILLMENT,
                        isDynamic: false
                    });
                    var lineCount = fulfillmentRecord.getLineCount('item');
                    for (var i = 0; i < lineCount; i++) {
                        fulfillmentRecord.setSublistValue('item', 'location', i, 1);
                    }
                })
            }
        }
    }
);
```

```

        var fulfillmentId = fulfillmentRecord.save();
        var so = record.load({
            type: record.Type.SALES_ORDER,
            id: result.id
        });
        so.setValue('memo', fulfillmentId);
        so.save();
        return true;
    });
} catch (e) {
    var subject = 'Fatal Error: Unable to transform salesorder to item fulfillment!';
    var authorId = -5;
    var recipientEmail = 'notify@company.com';
    email.send({
        author: authorId,
        recipients: recipientEmail,
        subject: subject,
        body: 'Fatal error occurred in script: ' + runtime.getCurrentScript().id +
'\n\n' + JSON.stringify(e)
    });
}
return {
    execute: execute
};
});

```

Scheduled Script Execution

When you schedule a script, you set the deployment time and not the execution time. After a script is deployed to the queue, it is executed serially (first in, first out) on a per-company basis. By default, there is a single queue used by all scheduled scripts in your company's NetSuite account. If you have a SuiteCloud Plus license, you can run scheduled scripts in parallel.

Execution of a scheduled script depends on how many scripts are ahead of it in the queue and how long it takes to run them. As soon as one script completes, the next script in the queue can be executed. Although multiple scheduled scripts can exist in a queue, only a single script can be executed at any specific time per queue.



Important: There may be a short system delay before your script is executed, even when there are no scripts currently in the scheduling queue.

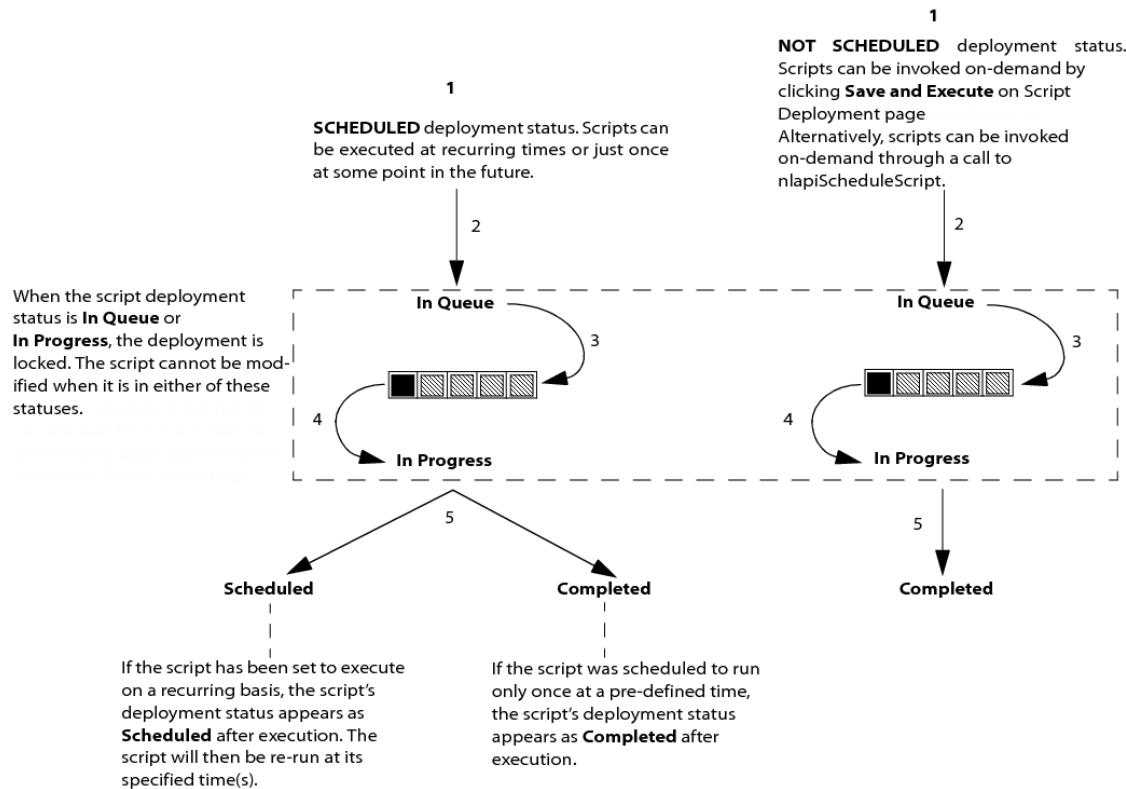
Scheduled Script Deployment

You can deploy a script to the scheduling queue for ad-hoc or future execution. For more information, see [Ad-hoc Deployment](#) and [Scheduled Deployment](#).

Be aware that the times you set on the Schedule subtab set the deployment time and not the execution time. If you do not have a SuiteCloud Plus license, and there are scripts already in the queue waiting to be executed, the script deployed into the queue is executed after all other scripts have completed. If no scripts are before it in the queue, there may be a short system delay before the script is executed.

Scheduled Scripts have two possible execution flows:

A script's **deployment** status set to **Scheduled** or **Not Scheduled** determines its path of execution in NetSuite.



Ad-hoc Deployment

To initiate an ad-hoc deployment of a script into the scheduling queue:

Before you begin, ensure that the SuiteScript file is uploaded and there is a script record for your script.

You can create single or multiple deployments for a script.

Note: If you set the deployment status to **Not Scheduled**, the settings on the Schedule tab are not applied. The script is only deployed as a single event.

Note: If you set the deployment status to **Testing**, the only way to place the script into the scheduling queue is by clicking **Save and Execute** on the Script Deployment page. You cannot schedule testing times and then click **Save**.

1. View the script record for your scheduled script by going to Customization > Scripting > Scripts.
2. Click Deploy Script.
3. Select the **Deployed** check box if it is not already checked.
4. From the **Status** field, do one of the following:

- To load a script into the SuiteScript Debugger, or to run the script only in the script owner's account, select **Testing**.
- To disable the scheduling options and manually trigger a scheduled script, select **Not Scheduled**.

5. Click the **Single Event** radio button.

The screenshot shows the 'Script Deployment' page. The 'SCRIPT' field contains 'Test Scheduled'. The 'TITLE' field is 'Test Scheduled 2'. The 'ID' field is empty. The 'STATUS' dropdown is set to 'Not Scheduled'. The 'LOG LEVEL' dropdown is set to 'Audit'. The 'EXECUTE AS ROLE' dropdown is set to 'Administrator'. Under the 'Schedule' tab, the 'SINGLE EVENT' radio button is selected. The 'START DATE' is '6/20/2016', 'START TIME' is '6:00 pm', and 'REPEAT' is set to 'No End Date'.

6. Click **Save**.

7. If the status was set to Testing, from the Script Deployment record, select Edit. Select Save and Execute.

The screenshot shows the 'Script Deployment' page with the 'Actions' dropdown open. The 'Save and Execute' option is highlighted with a blue box. The 'Status' dropdown is set to 'Not Scheduled'. The 'LOG LEVEL' dropdown is set to 'Debug'. The 'EXECUTE AS ROLE' dropdown is set to 'Administrator'.

Scheduled Deployment

You can deploy a script to the scheduling queue at a pre-defined time in the future. For example, a script that does not need to be executed immediately, or a script that was set to Testing and is now ready for production.

You can also define a recurrence and frequency for multiple deployments. With a deployment status set to Scheduled, the script will run again at the next scheduled time.



Important: If the deployment status of a script is set to Scheduled, you cannot programmatically put the script into the scheduling queue.

To initiate a scheduled deployment of a script:

1. Go to Customization > Scripting > Scripts and view the script record for your scheduled script.
2. Click Deploy Script.
3. Select the **Deployed** check box if it is not already checked.
4. From the **Status** field, select Scheduled.
5. On the **Schedule** tab, set all deployment options.
6. Click **Save**.

Multiple Deployments

To edit or access each deployment, go to Customization > Scripting > Script Deployments.

To schedule multiple deployments for the same script:

1. View a script deployment record for your scheduled script by going to Customization > Scripting > Script Deployments.
2. On the Script deployment record page, click Actions > New or Make Copy.

The screenshot shows a 'Script Deployment' page. At the top, there are buttons for 'Edit' and 'Back'. To the right of 'Edit' is a dropdown menu labeled 'Actions' with two options: 'New' and 'Make Copy'. Below this, there is a section for 'SCRIPT' with the value 'SSTF Bundle Tests Runner'. Under 'TITLE', the value is 'SSTF Bundle Tests Runner Secondary'. At the bottom, there is an 'ID' field.

3. Optionally, create your own unique deployment ID in the **ID** column. If you do not add your own custom deployment ID, an ID is automatically generated.
4. When finished creating all deployments, click **Save**.

Scheduled Script Status and Monitoring

Runtime Status

To access the current and past runtime statuses of all scheduled scripts that have been executed in your account, go to Customization > Scripting > Scheduled Script Status.

To access the runtime statuses of all deployed instances of a particular script, go to a Script Deployment page and click See Instances.



Important: Script execution details are purged after 30 days.

Pending	The script is in the queue and ready to be processed.
---------	---

Deferred	The script is eligible for processing but has not been processed due to constraints. For example, deferred status will occur when one task must wait for another to finish.
Processing	The script is running.
Retry	The script entered the processing state and failed to complete normally. It is eligible to be retried. There is no need to create new deployment as the script will be retried automatically. To help determine the failure reason (for example, timeout problem), see the script log.
Complete	The script completed normally.
Cancelled	Due to a NetSuite server error, the script was cancelled during or before script execution.
Failed	The script began processing but failed to complete normally. Examine your script for possible errors.

execute

Description	Definition of the scheduled script trigger point
Returns	void
Since	Version 2015 Release 2

Parameters

Note: The `scriptContext` parameter is a JavaScript object. NetSuite automatically passes this object to the script entry point.

Parameter	Type	Required / Optional	Description	Since
<code>scriptContext.type</code>	string	required	The context in which the script is executed. Values are reflected in the <code>context.InvocationType</code> enum.	Version 2015 Release 2

context.InvocationType

Enum Description	Enumeration that holds the string values for scheduled script execution contexts.
Module	Scheduled Script Type
Since	Version 2015 Release 2

Values

SCHEDULED	The normal execution according to the deployment options specified in the UI.
ON_DEMAND	The script is executed via a call from a script (using ScheduledScriptTask.submit()).
	<p> Note: The scheduled script must have a status of Not Scheduled on the Script Deployment page.</p>
USER_INTERFACE	The script is executed via the UI (the Save & Execute button has been clicked).
ABORTED	The script re-executed automatically following an aborted execution (system went down during execution).
SKIPPED	The script is executed automatically following downtime during which the script should have been executed.

Best Practice: Handling Server Restarts in Scheduled Scripts (SuiteScript 2.0)

Occasionally, a scheduled script failure may occur due to an application server restart. This could be due to a NetSuite update or maintenance, or an unexpected failure of the execution environment. Restarts can terminate an application forcefully at any moment. Therefore, robust scripts need to account for restarts and be able to recover from an unexpected interruption.

In SuiteScript 2.0, in the event of an unexpected system failure, the script is restarted from the beginning. The NetSuite system can automatically detect when a scheduled script is forcefully terminated, and restart the script as soon as the resources required to run the script are available. Note that in SuiteScript 2.0, yields and recovery points are not manually scripted. Therefore, handling a restart situation is simpler.

The following sample scripts demonstrate how restarts impact scheduled script execution:

- [Simple Scheduled Script \(SuiteScript 2.0\)](#)
- [Example: A Problematic Scheduled Script](#)
- [Example: A Robust Scheduled Script](#)

Simple Scheduled Script (SuiteScript 2.0)

If there is a forceful termination in any part of the script, the script is always restarted from the beginning.

The script can detect a restarted execution by examining the `type` attribute of the `context` argument. The script is being restarted if the value of this argument is (`context.type === "aborted"`). For more information about the `context` argument, see [execute context.InvocationType](#).

```
/** 
 * @NApiVersion 2.0
 * @NScriptType scheduledscript
 */
define([], function(){
```

```

        return {
            execute: function (context)
            {
                if (context.type === 'aborted')
                {
                    // I might do something differently
                }
                .
                .
                .
            }
        });
    });
});

```

Example: A Problematic Scheduled Script

A very common pattern seen in scheduled scripts is to perform a search and then processing the results. Consider the following script:

The purpose of this example script is to update each sales order in the system. The `filter1` filter ensures that the search returns exactly one entry per sales order. However, if the script is forcefully interrupted during its processing and then restarted, some sales orders might be updated twice.

To prevent data issues that result from re-processing, the script should use idempotent operations. This means that any operation handled by the script can repeat itself with the same result (e.g. prevent creating duplicate records). Or, the script must be able to recover (e.g. by creating a new task to remove duplicates).

This script does not use idempotent operations, and a large number of sales orders could be updated twice (for example, doubling a price on a sales order from a repeated operation). To improve a script like this, see [Example: A Robust Scheduled Script](#) and [Example 5: A Robust Map/Reduce Script](#).

```

/**
 * @NApiVersion 2.0
 * @NScriptType scheduledscript
 */
define(['N/search', 'N/record'],
    function(search, record){
        return {
            execute: function (context)
            {
                var filter1 = search.createFilter({
                    name: 'mainline',
                    operator: search.Operator.IS,
                    values: true
                });
                var srch = search.create({
                    type: search.Type.SALES_ORDER,
                    filters: [filter1],
                    columns: []
                });

                var pagedResults = srch.runPaged();
            }
        });
    });
});

```

```

pagedResults.pageRanges.forEach(function(pageRange){
    var currentPage = pagedResults.fetch({index: pageRange.index});
    currentPage.data.forEach(function(result){
        var so = record.load({
            type: record.Type.SALES_ORDER,
            id: result.id
        });
        //UPDATE so FIELDS>
        so.save()
    });
});
}
);
});
});
```

Example: A Robust Scheduled Script

The following sample code uses `custbody_processed_flag` on the sales order. It is a custom boolean field that must be previously created in the UI. When the sales order is processed, the field is set to `true`. The search then contains an additional filter that excludes flagged sales orders. When the script is restarted, only the sales order which have not been updated are processed.

```

/**
 * @NApiVersion 2.0
 * @NScriptType scheduledscript
 */
define(['N/search', 'N/record'],
    function(search, record){
    return {
        execute: function (context)
        {
            var filter1 = search.createFilter({
                name: 'mainline',
                operator: search.Operator.IS,
                values: true
            });
            var filter2 = search.createFilter({
                name: 'custbody_processed_flag',
                operator: search.Operator.IS,
                values: false
            });
            var srch = search.create({
                type: search.Type.SALES_ORDER,
                filters: [filter1, filter2],
                columns: []
            });

            var pagedResults = srch.runPaged();

            pagedResults.pageRanges.forEach(function(pageRange){
                var currentPage = pagedResults.fetch({index: pageRange.index});
                currentPage.data.forEach(function(result){
                    var so = record.load({
                        type: record.Type.SALES_ORDER,
```

```
        id: result.id
    });
    // UPDATE so FIELDS
    so.setValue({
        fieldID: 'custbody_processed_flag',
        value: true
    });
    so.save()
});
});
```

Suitelet Script Type

Suitelets are extensions of the SuiteScript API that allow you to build custom NetSuite pages and backend logic. Suitelets are server-side scripts that operate in a request-response model, and are invoked by HTTP GET or POST requests to system generated URLs.

There are two types of Suitelets:

1. **Suitelets** use UI objects to create custom pages that look like NetSuite pages. SuiteScript UI objects encapsulate the elements for building NetSuite-looking portlets, forms, fields, sublist, tabs, lists, and columns.
 2. **Backend Suitelets** do not use any UI objects and execute backend logic, which can then be parsed by other parts of a custom application. Backend Suitelets are best used for the following purposes:
 - Providing a service for backend logic to other SuiteScripts, or to other external hosts outside of NetSuite.
 - Offloading server logic from client scripts to a backend Suitelet shipped without source code to protect sensitive intellectual property.



Important: RESTlets can be used as an alternative to backend Suitelets.



Note: Suitelets are not intended to work inside web stores. Use online forms to embed forms inside a web store.

How Suitelet Scripts are Executed

The following steps and diagram provide an overview of the Suitelet execution process:

1. Client initiates an HTTP GET or POST request (typically from a browser) for a system-generated URL. A web request object contains the data from the client's request.
 2. The user's script is invoked, which gives the user access to the entire Server SuiteScript API as well as a web request and web response object.
 3. NetSuite processes the user's script and returns a web response object to the client. The response can be in following forms:
 - Free-form text
 - HTML

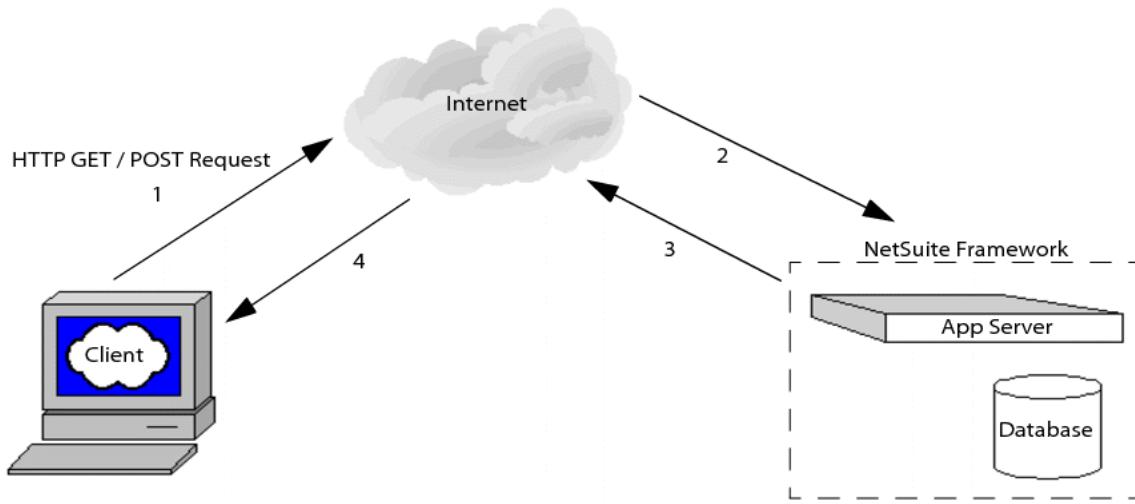
- RSS
- XML
- A browser redirect to another page on the Internet



Important: You can only redirect to external URLs from Suitelets that are accessed externally (in other words, the Suitelet has been designated as "Available Without Login" and is accessed from its external URL).

- A browser redirect to an internal NetSuite page. The NetSuite page can be either a standard page or custom page that has been dynamically generated using UI objects.

4. The data renders in the user's browser.



Reserved Parameter Names in Suitelet URLs

Certain names are reserved and should not be referenced when naming custom parameters for Suitelet URLs.

The following table contains a list of reserved parameter names:

Reserved Suitelet URL Parameter Names

e	print
id	email
cp	q
l	si
popup	st
s	r
d	displayonly
_nodrop	nodisplay
sc	deploy
sticky	script

If any of your parameters are named after any of the reserved parameter names, your Suitelet may throw an error saying, "There are no records of this type." To avoid naming conflicts, NetSuite recommends that all custom URL parameters are prefixed with **custom**. For example, use **custom_id** instead of **id**.

Best Practices

The following list shows best practices for both form-level and record-level client script development:

- Suitelets are ideal for generating NetSuite pages (forms, lists), returning data (XML, text), and redirecting requests.
- Limit the number of UI objects on a page (< 100 rows for sublists, < 100 options for ad-hoc select fields, < 200 rows for lists).
- Experiment with inline HTML fields embedded on form before going the full custom HTML page route.
- Deploy Suitelets as “Available without Login” only if absolutely necessary (no user context, login performance overhead). (See the help topic [Setting Available Without Login](#).)
- Append “ifrmcntnr=T” to the external URL when embedding in iFrame especially if you are using Firefox. (For more about NetSuite and iFrame, see the help topic [Embedding an Online Form in your Web Site Page](#).)
- When building custom UI outside of the standard NetSuite UI (such as building a custom mobile page using Suitelet), use the User Credentials APIs to help users manage their credentials within the custom UI. For more information, see [User Credentials APIs](#).

Suitelet Script Type Sample

The following sample shows how to create a Suitelet form that lets you write and send an email:

```
/**
 *@NApiVersion 2.x
 *@NScriptType Suitelet
 */
define(['N/ui/serverWidget', 'N/email', 'N/runtime'],
    function(ui, email, runtime) {
        function onRequest(context) {
            if (context.request.method === 'GET') {
                var form = ui.createForm({
                    title: 'Demo Suitelet Form'
                });
                var subject = form.addField({
                    id: 'subject',
                    type: ui.FieldType.TEXT,
                    label: 'Subject'
                });
                subject.layoutType = ui.FieldLayoutType.NORMAL;
                subject.breakType = ui.FieldBreakType.STARTCOL;
                subject.isMandatory = true;
                var recipient = form.addField({
                    id: 'recipient',
                    type: ui.FieldType.EMAIL,
                    label: 'Recipient email'
                });
                recipient.isMandatory = true;
                var message = form.addField({
                    id: 'message',
                    type: ui.FieldType.TEXTAREA,
                    label: 'Message'
                });
            }
        }
    }
);
```

```

    });
    message.displaySize = {
        width: 60,
        height: 10
    };
    form.addSubmitButton({
        label: 'Send Email'
    });
    context.response.writePage(form);
} else {
    var request = context.request;
    email.send({
        author: runtime.getCurrentUser().id,
        recipients: request.parameters.recipient,
        subject: request.parameters.subject,
        body: request.parameters.message
    });
}
return {
    onRequest: onRequest
};
});

```

onRequest(params)

Description	Definition of the Suitelet script trigger point.
Returns	void
Since	Version 2015 Release 2

Parameters

Note: The `params` parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
<code>params.request</code>	<code>http.ServerRequest</code>	required	The incoming request.	Version 2015 Release 2
<code>params.response</code>	<code>http.ServerResponse</code>	required	The Suitelet response.	Version 2015 Release 2

User Event Script Type

User event scripts are executed on the NetSuite server. They are executed when users perform certain actions on records, such as create, load, update, copy, delete, or submit. Most standard NetSuite records and custom record types support user event scripts.

User event scripts can be used to perform the following tasks:

- Implement custom validation on records.
- Enforce user-defined data integrity and business rules.
- Perform user-defined permission checking and record restrictions.
- Implement real-time data synchronization.
- Define custom workflows. (redirection and follow-up actions)
- Customize forms.



Important: User event scripts cannot be executed by other user event scripts or by workflows with a **Context of User Event Script**. In other words, you cannot chain user event scripts. You can, however, execute a user event script from a call within a scheduled script, a portlet script, or a Suitelet.

How User Events are Executed

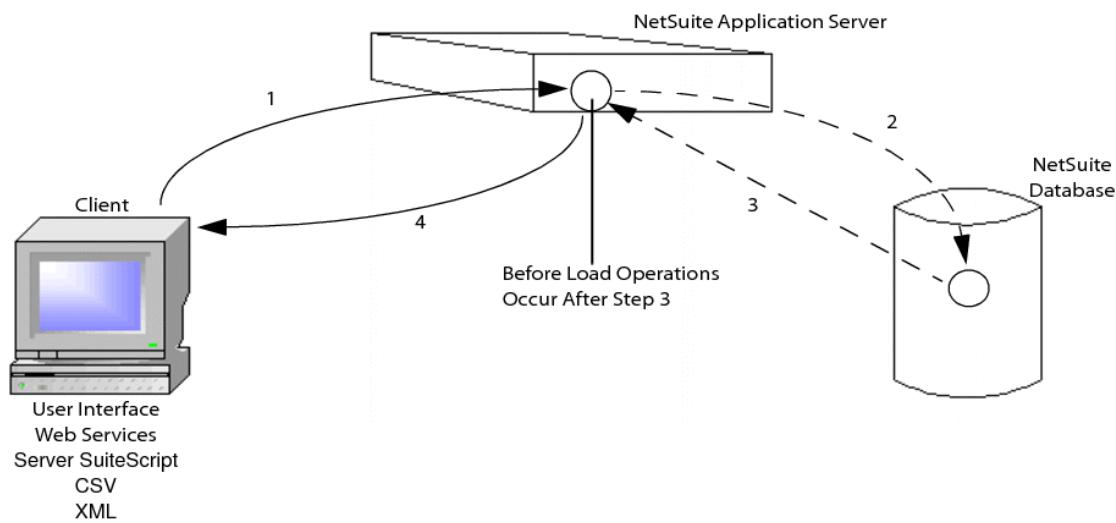
User event scripts are executed based on operation types defined as beforeLoad, beforeSubmit, and afterSubmit.

The following steps and diagram provide an overview of what occurs during a beforeLoad operation:

1. The client sends a read operation request for record data. A client request can come from the user interface, web services, server-side SuiteScript calls, CSV imports, or XML.
2. Upon receiving the request, the application server performs basic permission checks on the client.
3. The database loads the requested information into the application server for processing. This is where the beforeLoad operation occurs – before the requested data is returned to the client.
4. The client receives the now validated/processed beforeLoad data.



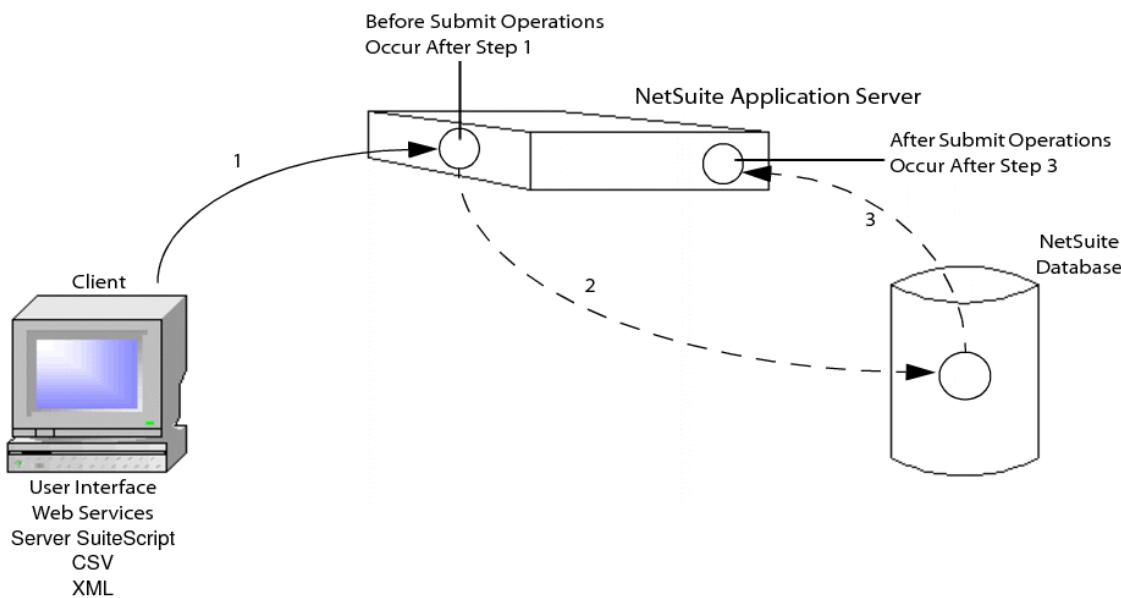
Note: Standard records cannot be sourced during a beforeLoad operation. Use the pageInit client script for this purpose. See [pageInit](#).



The following steps and diagram provide an overview of what occurs on beforeSubmit and afterSubmit operations:

1. The client performs a write operation by submitting data to the application server. (The client request can come from the user interface, web services, server-side SuiteScript calls, CSV imports, or XML.) The application server:
 - a. performs basic permission checks on the client
 - b. processes the submitted data and performs specified validation checks during a beforeSubmit operation
- The submitted data has **NOT** yet been committed to the database.
2. After the data has been validated, it is committed to the database.
 3. If this (newly committed) data is then called by an afterSubmit operation, the data is taken from the database and is sent to the application server for additional processing. Examples of afterSubmit operations on data that are already committed to the database include, but are not limited to:
 - a. sending email notifications (regarding the data that was committed to the database)
 - b. creating child records (based on the data that was committed to the database)
 - c. assigning tasks to employees (based on data that was committed to the database)

Note: Asynchronous afterSubmit user events are only supported during webstore checkout.



Best Practices

The following list shows best practices for user event script development:

- Use the type argument and context object to define and limit the scope of your user event logic.
- Limit the amount of script execution in user event scripts (< 5 seconds) since they run often and inline. You can use the Script Performance Monitor SuiteApp to test the performance of your scripts deployed on a specific record type.
- Mission critical business logic implemented using user events should be accompanied by a 'Clean up' scheduled script to account for any unexpected errors or mis-fires.

- Any operation that depends on the submitted record being committed to the database should happen in an afterSubmit script.
- Be careful when updating transaction line items in a beforeSubmit script because you have to ensure that the line item totals net taxes and discounts are equal to the summarytotal, discounttotal, shippingtotal, and taxtotal amounts.
- Activities (user events) on a hosted Web site can trigger server-side SuiteScripts. In addition to Sales Orders, scripts on Case and Customer records also execute because of Web activities.
- Assigning many executable functions to one record type is discouraged because this could negatively affect the user experience with that record type. For example, if there are ten beforeLoad scripts that must complete their execution before the record loads into the browser, the time needed to load the record may increase significantly. Be aware of the number of user events scripts used, including bundled user event scripts.
- To set a field on a record or make any changes to a record being submitted, use the beforeSubmit event rather than the afterSubmit event. Changes made during an afterSubmit event duplicates the record.
- Perform all post-processing operations of the current record on an afterSubmit event.

User Event Script Sample

The following sample shows a user event script. This script is designed for use in environments that do not use the Team Selling feature.

When you deploy this script on the customer record, this script creates a follow-up phone call record for every newly created customer record.



Important: Before running this script, you must replace the salesrep internal ID with one specific to your account. Specifically, use an ID that represents an employee who is classified as a sales rep. The sales rep option is located on the Human Resources subtab of the employee record. If you do not replace the ID, the script may not work as expected. Additionally, note that this script is designed to work in environments where the customer record includes a salesrep field. If the Team Selling feature is enabled, the customer record typically will not include a salesrep field.

```
/** 
 * @NApiVersion 2.x
 * @NScriptType UserEventScript
 */
define(['N/record'],
  function(record) {
    function beforeLoad(context) {
      if (context.type !== context.UserEventType.CREATE)
        return;
      var customerRecord = context.newRecord;
      customerRecord.setValue('phone', '555-555-5555');
      if (!customerRecord.getValue('salesrep'))
        customerRecord.setValue('salesrep', 46);
    }
    function beforeSubmit(context) {
      if (context.type !== context.UserEventType.CREATE)
        return;
      var customerRecord = context.newRecord;
```

```

        customerRecord.setValue('comments', 'Please follow up with this customer!');
    }
    function afterSubmit(context) {
        if (context.type !== context.UserEventType.CREATE)
            return;
        var customerRecord = context.newRecord;
        if (customerRecord.getValue('salesrep')) {
            var call = record.create({
                type: record.Type.PHONE_CALL,
                isDynamic: true
            });
            call.setValue('title', '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('Call record created successfully', 'Id: ' + callId);
            } catch (e) {
                log.error(e.name);
            }
        }
    }
    return {
        beforeLoad: beforeLoad,
        beforeSubmit: beforeSubmit,
        afterSubmit: afterSubmit
    };
});

```

User Event Script API

Script Entry Point	Description
afterSubmit(scriptContext)	Executed immediately after a write operation on a record
beforeLoad(scriptContext)	Executed whenever a read operation occurs on a record, and prior to returning the record or page.
beforeSubmit(scriptContext)	Executed prior to any write operation on the record.
context.UserEventType	Holds the string values for user event execution contexts.

afterSubmit(scriptContext)

Description	Executed immediately after a write operation on a record. This event can be used to include email notification, re-direct the browser, create dependent records, and synchronize with an external system.
Returns	void

Since	Version 2015 Release 2
-------	------------------------

Parameters

i Note:	The scriptContext parameter is a JavaScript object.
----------------	---

Parameter	Type	Required / Optional	Description	Since
scriptContext.newRecord	record.Record	required	The new record	Version 2015 Release 2
scriptContext.oldRecord	record.Record	required	The old record	Version 2015 Release 2
scriptContext.type	string	required	The trigger type. Use the context.UserEventType enum to set this value.	Version 2015 Release 2

beforeLoad(scriptContext)

Description	Executed whenever a read operation occurs on a record, and prior to returning the record or page. These operations include navigating to a record in the UI, reading a record in web services, and loading a record. This event cannot be used to source standard records. Use the pageInit client script for this purpose.
i Note:	beforeLoad user events cannot be triggered when you load/access an online form.
Returns	void
Since	Version 2015 Release 2

Parameters

i Note:	The scriptContext parameter is a JavaScript object.
----------------	---

Parameter	Type	Description	Since
scriptContext.newRecord	record.Record	The new record.	Version 2015 Release 2
scriptContext.type	string	The type of operation invoked by the event. The type can be any of the possible values for the context.UserEventType enum. Type arguments vary depending on whether the event	Version 2015 Release 2

Parameter	Type	Description	Since
		occurs during a beforeLoad, beforeSubmit, or afterSubmit operation. This parameter allows the script to branch out to different logic depending on the operation type. For example, a script with deltree logic that deletes a record and all of its child records should only be invoked when type equals <code>delete</code> . User event scripts should always check the value of the type argument to avoid indiscriminate execution.	
<code>scriptContext.form</code>	<code>serverWidget.Form</code>	The current form.	Version 2015 Release 2
<code>scriptContext.request</code>	<code>http.ServerRequest</code>	The HTTP request information sent by the browser. If the event was triggered by a server action, this value is not present.	Version 2015 Release 2

beforeSubmit(scriptContext)

Description	Executed prior to any write operation on the record. Changes made to the current record in this script persist after the write operation. This event can be used to validate the submitted record, perform any restriction and permission checks, and perform any last-minute changes to the current record.
Returns	<code>void</code>
Since	Version 2015 Release 2

Parameters

 **Note:** The `scriptContext` parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
<code>scriptContext.newRecord</code>	<code>record.Record</code>	required	The new record.	Version 2015 Release 2
<code>scriptContext.oldRecord</code>	<code>record.Record</code>	required	The old record.	Version 2015 Release 2
<code>scriptContext.type</code>	<code>string</code>	required	The trigger type. Use the <code>context.UserEventType</code> enum to set this value.	Version 2015 Release 2

context.UserEventType

Enum Description	Holds the string values for user event execution contexts.
Module	User Event Script Type
Since	Version 2015 Release 2

Values

- APPROVE
- CANCEL
- CHANGEPASSWORD
- COPY
- CREATE
- DELETE
- DROPSHIP
- EDIT
- EDITFORECAST
- EMAIL
- MARKCOMPLETE
- ORDERITEMS
- PACK
- PAYBILLS
- PRINT
- QUICKVIEW
- REASSIGN
- REJECT
- SHIP
- SPECIALORDER
- TRANSFORM
- VIEW
- XEDIT



Note: Attaching a child custom record to its parent or detaching a child custom record from its parent will trigger an edit event.

Workflow Action Script Type

Workflow action scripts allow you to create custom Workflow Actions that are defined on a record in a workflow. Workflow action scripts are useful for performing actions on sublist because sublist fields are not currently available through the Workflow Manager. Workflow action scripts are also useful

when you need to create custom actions that execute complex computational logic that is beyond what can be implemented with the built-in actions.

Workflow Action Script Sample

The sample script is designed to execute each time an Expense Report record is created. The script counts the number of expense items submitted for the employee in the past month. Based on this value and a threshold value supplied by a script parameter in NetSuite, the code then updates the Comments field of the Employee record to indicate whether the employee is a frequent traveller.

This script sample assumes the following set-up in the NetSuite account:

- In the Script record for the sample script, a parameter with the Label field set to Frequent Travel Limit, the Type field set to Integer Number type, and the ID set to custscript_sdr_freq_trav_limit is added. The Return Type field is set to Free-Form Text.
- In the Script record for the sample script, the deployment applies to Expense Report records.
- In a new Workflow, the Record Type field is set to Transaction, and the Sub Types field is set to Expense Report. The Workflow Release Status field is set to Testing.
- In a new Workflow field, set the Label field to Status.
- The Start State contains a new Action set to the deployed Workflow Action script. In the Action, set the Store Result In field to Status (Workflow). Set the Frequent Travel Limit value field to 2.

```
/** 
 * @NApiVersion 2.x
 * @NScriptType WorkflowActionScript
 */
define([], 
    function() {
        function onAction(scriptContext) {
            log.debug({
                title: 'Start Script'
            });
            var newRecord = scriptContext.newRecord;
            var itemCount = newRecord.getLineCount({
                sublistId: 'item'
            });
            log.debug({
                title: 'Item Count',
                details: itemCount
            });
            for(var i = 0; i < itemCount; i++)
            {
                var quantity = newRecord.getSublistValue({
                    sublistId: 'item',
                    fieldId: 'quantity',
                    line: i
                });
                log.debug({
                    title: 'Quantity of Item ' + i,
                    details: quantity
                });
                if(quantity === 0)
                {
```

```

        return 0;
    }
}
log.debug({
    title: 'End Script'
});
return 1;
}
return {
    onAction: onAction
}
});

```

onAction(scriptContext)

Description	Defines a Workflow Action script trigger point.
Returns	void
Since	Version 2016 Release 1

Parameters

Note: The `scriptContext` parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
<code>scriptContext.newRecord</code>	<code>record.Record</code>	required	The new record. <code>record.Record.save()</code> is not permitted.	2016.1
<code>scriptContext.oldRecord</code>	<code>record.Record</code>	required	The old record. <code>record.Record.save()</code> is not permitted.	2016.1
<code>scriptContext.form</code>	<code>serverWidget.Form</code>	optional	The form through which the script interacts with the record. This parameter is available only in the <code>beforeLoad</code> context.	2016.2
<code>scriptContext.type</code>	<code>string</code>	optional	An event type, such as <code>create</code> , <code>edit</code> , <code>view</code> , or <code>delete</code> .	2016.2
<code>scriptContext.workflowId</code>	<code>integer</code>	optional	The internal ID of the workflow that calls the script.	2016.2

SuiteScript 2.0 Global Objects and Methods

SuiteScript 2.0 includes the following global objects and methods. You are not required to load them.

- [define Function](#)
- [require Function](#)
- [log Object](#)
- [util Object](#)
- [toString\(\)](#)
- [JSON object](#)
- [Promise object](#)

Module Dependency Paths

To specify module paths and module names, you can use the [define Function](#) or the [require Function](#).

Avoid using any file extensions in the path.

SuiteScript supports the following:

Path Type	Description
Absolute Paths	Specifies the path from the root folder in the file cabinet. Absolute paths start with a forward slash (/). For example, "/SuiteScripts/MyApp/Util".
Relative Paths	Specifies the path relative to the dependent module's location in the file cabinet. This provides greater flexibility when moving nested folders containing modules in the file cabinet. Relative Paths start with a period (.). For example, assume that the dependent module is located in the following folder: "/SuiteScripts/MyApp". The relative path for a sibling file under "SuiteScripts/MyApp" could be "./Util". The equivalent absolute path in this case would be "/SuiteScripts/MyApp/Util".
<div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> Note: Do not use relative paths in global contexts. For example, when using the require function or SuiteScript Debugger, . </div>	
Bundle Virtual Paths	Specifies a bundle path that the NetSuite system can resolve to a file cabinet pointer. A valid bundle ID is required in the bundle virtual path. Bundle virtual paths start with a forward slash followed by a period (/.). For example: <ul style="list-style-type: none"> ▪ ./bundle/<bundle id>/SS2_CustomModuleTest.js/ ▪ ./suiteapp/100
Naming a Custom Module	Defines a global identifier used to reference a module by name and not path. Use a require Configuration to set the name.

Absolute Paths

Use an initial forward slash (/) to denote the top-level File Cabinet directory.

In the following example, the Module Loader expects the custom module files `lib1.js` and `lib2.js` to be located in the `SuiteScripts` top-level directory in the File Cabinet.

```
// myModule.js
define(['./SuiteScripts/lib1', './SuiteScripts/lib2'], // myModule has a dependency on modules lib1 and lib2
      ...
);
```

Relative Paths

You can specify relative paths to modules in subdirectories from the directory of the current module.

In the following example, the Module Loader expects the custom module files `lib1.js` and `lib2.js` to be located in the `lib` subdirectory, the same File Cabinet directory that contains `myModule.js`.

```
// myModule.js
define ['./lib/lib1', './lib/lib2'], // myModule has a dependency on modules lib1 and lib2
      ...
);
```

Bundle Virtual Paths

Use a bundle virtual path to point to a SuiteApp file cabinet location. Bundle virtual paths start with a forward slash followed by a period (/.). A valid bundle ID is required in the bundle virtual path.

In the following example, the Module Loader is looking for common libraries in shared bundles.

```
// myModule.js
require(['/.suiteapp/100','/.bundle/101'], // myModule has a dependency on modules with the bundle id 100 and 101
      ...
);
```

In certain situations, a virtual path can help when a SuiteApp is moved or the bundle id changes due to deprecation or copying. At the time of copying or deprecation, SuiteScript 2.0 checks the deprecation or copy chain and looks for the file in multiple places. For example, a created bundle is deprecated by a newer version of the bundle, and the newer version is installed in the target account with a different ID. In this case, the scripts using the virtual path of the deprecated bundle in the target account use the new bundle's ID, because the virtual bundle path automatically resolves to the new bundle.

Be aware that the ID for the created bundle in the source account could not be used to specify a virtual bundle path. The virtual bundle path depends on a file path to an existing bundle. At the time the bundle is created in the source account, no such valid file path is created yet.

define Function

The `define` Function is a global object that implements a `define()` Module Loader interface for SuiteScript 2.0. It conforms to the Asynchronous Module Definition (AMD) specification.

This function is overloaded. SuiteScript supports the following method signatures:

Type	Name	Return Type / Value Type	Description
Function	define(moduleobject)	Void	Defines a SuiteScript 2.0 entry point script or a SuiteScript 2.0 custom module. Returns the module object defined by the <code>moduleobject</code> parameter.
	define(id, [dependencies,] callback)	Void	Defines a SuiteScript 2.0 entry point script or a SuiteScript 2.0 custom module. Loads the dependencies and then executes the callback function.
	define([dependencies,] callback)	Void	Defines a SuiteScript 2.0 entry point script or a SuiteScript 2.0 custom module. Loads the dependencies and then executes the callback function.
	define(callback)	Void	Defines a SuiteScript 2.0 entry point script or a SuiteScript 2.0 custom module if no dependencies are required. Executes the callback function.

The `define()` Function works as a factory function for SuiteScript 2.0 script types and custom SuiteScript 2.0 modules. This function executes asynchronously on the client side and synchronously on the server side.

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. You can use the `define([dependencies,] callback)` or `define(id, [dependencies,] callback)` signature. If the custom module does not require any dependencies, you can use the `define(moduleobject)` or `define(callback)` signature.

For more information about custom modules, see [SuiteScript 2.0 Custom Modules](#).

For more information about entry points, see [SuiteScript 2.0 Script Types and Entry Points](#).

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 ad-hoc debug your code 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 HTTPS.

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

define([dependencies,] callback)

Method Description	Function used to define a SuiteScript 2.0 entry point script or a SuiteScript 2.0 custom module. When NetSuite executes the function, the Module Loader loads the dependencies and then executes the callback function. It keeps the callback return value to be provided to other modules asking for it as a dependency. When setting the callback parameter value, in place of the callback function, you can directly pass in a module object (the callback return value). If defining an entry point script, the return statement must return at least one entry point and entry point function. This function conforms to the Asynchronous Module Definition (AMD) specification. For more information, see define Function and SuiteScript 2.0 – Script Architecture .
Returns	Void
Supported Script Types	All script types
Governance	None
Global object	define Function
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
dependencies	string []	Optional	<p>Represents any module dependencies required by the callback function. Use the following syntax:</p> <ul style="list-style-type: none"> ■ Native SuiteScript 2.0 modules: ['N/<module name>'] ■ Custom modules: [/<path to module file in File 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			Cabinet>/<module name>'] For other options, see Module Dependency Paths .	
callback	Function Object	Required	Callback function to execute after dependent modules are loaded.	Version 2015 Release 2

Errors

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

Syntax for SuiteScript 2.0 Script Type

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 convertT
oThing()
        }
    }
});

```

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 mo
dule
});

```

For more information about custom modules, see [SuiteScript 2.0 Custom Modules](#).

define(id, [dependencies,] callback)

Method Description	Function used to define a SuiteScript 2.0 entry point script or a SuiteScript 2.0 custom module. When NetSuite executes the function, the Module Loader loads
---------------------------	---

	the dependencies and then executes the callback function. It keeps the return value to be provided to other modules asking for it as a dependency. When setting the callback parameter value, in place of the callback function, you can directly pass in a module object (the callback return value). If defining an entry point script, the return statement must return at least one entry point and entry point function. This function conforms to the Asynchronous Module Definition (AMD) specification. For more information, see define Function and SuiteScript 2.0 – Script Architecture .
Returns	Void
Supported Script Types	All script types
Governance	None
Global object	<code>define Function</code>
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
<code>id</code>	string	optional	Defines the id of the module	Version 2015 Release 2
<code>dependencies</code>	string []	optional	<p>Represents any module dependencies required by the callback function. Use the following syntax:</p> <ul style="list-style-type: none"> ■ Native SuiteScript 2.0 modules: ['N/<module name>'] ■ Custom modules: [/ '<path to module file in File Cabinet>/<module name>'] <p>For other options, see Module Dependency Paths.</p>	Version 2015 Release 2
<code>callback</code>	Function Object	required	Callback function to execute after dependent modules are loaded.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
<code>MODULE_DOES_NOT_EXIST</code>	Module does not exist: {module path/name}	The NetSuite module or custom module dependency does not exist. If multiple modules do not exist, NetSuite only reports the first error encountered. If

Error Code	Message	Thrown If
		you receive this error, verify that all module paths and names are correct.

Syntax

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){...});
...
```

define(callback)

Method Description	Function used to define a SuiteScript 2.0 entry point script or a SuiteScript 2.0 custom module. When NetSuite executes the function, the Module Loader executes the callback function. It keeps the return value to be provided to other modules asking for it as a dependency. If defining an entry point script, the return statement must return at least one entry point and entry point function. This function conforms to the Asynchronous Module Definition (AMD) specification. For more information, see define Function and SuiteScript 2.0 – Script Architecture .
Returns	Void
Supported Script Types	All script types
Governance	None
Global object	define Function
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
callback	Function	Required	Callback function to execute Must return entry points for SuiteScript 2.0 script types.	Version 2015 Release 2

Errors

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

Error Code	Message	Thrown If
		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.

Syntax

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

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

OR

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

define(moduleobject)

Method Description	Function used to define a SuiteScript entry point script or define a custom module if no dependencies are required. If defining an entry point script, the return statement must return at least one entry point and entry point function. For example, if you included a function in a file named <code>sample.js</code> and saved the file in the SuiteScripts File Cabinet folder, use the following syntax to require it as a dependency in other script files: <code>'/SuiteScripts/sample', function(sample) ...'</code>
Returns	Void
Supported Script Types	All script types
Governance	None
Global object	define Function
Since	Version 2015 Release 2

Parameters



Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
moduleobject	Object	Required	Any Javascript 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");
```

require Function

The require Function is a global object that implements the require() Module Loader interface for SuiteScript 2.0. It conforms to the Asynchronous Module Definition (AMD) specification. When NetSuite

executes the require() Function, it executes the callback function and loads the dependencies when they are needed.

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



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

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	require([dependencies,] callback)	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.
	require Configuration	require Function	Method reserved to configure loading for a custom module. Supports properties that can hold feature metadata, aliases, paths, package, and mapping information related to a module id.

require([dependencies,] callback)

Method Description	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.
>Returns	Void
Supported Script Types	All script types
Governance	None
Global object	require Function
Since	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"> ■ Native SuiteScript 2.0 modules: ['N/<module name>'] ■ Custom modules: ['/<path to module file in File Cabinet>/<module name>'] <p>See Module Dependency Paths.</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}	The NetSuite module or custom module dependency does not exist. If multiple modules do not exist, NetSuite only reports the first error encountered. If you receive this error, verify that all module paths and names are correct.

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

SuiteScript provides advanced options that provide you with greater control over require configuration. You can use an advanced API ([require.config\(\)](#)) to implement require configuration values. For a list of supported configuration parameters, see [require Configuration Parameters](#).

require Configuration Parameters

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

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

Parameter	Type	Required / Optional	Sample Usage	Since
config	Object	Optional	<ul style="list-style-type: none"> ■ To assign attributes, such as metadata. 	Version 2015 Release 2
shim	Object	Optional	<ul style="list-style-type: none"> ■ To prepare a non-AMD JS library for loading. 	Version 2015 Release 2

require.config()

Method Description	Function used to set properties before loading a custom module. For example, to set up configuration for improved compatibility. When you use this method, it should precede the module definition. This function conforms to the Asynchronous Module Definition (AMD) specification. For more information about this method and its parameters, see https://github.com/amdjs/amdjs-api/blob/master/CommonConfig.md (version fd45c71). Also see require Configuration Parameters .
Returns	require Function
Supported Script Types	Server-side scripts
Governance	None
Global object	require Function
Since	Version 2015 Release 2

log Object

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

log Object Members

- [log.debug\(options\)](#)
- [log.audit\(options\)](#)
- [log.emergency\(options\)](#)
- [log.error\(options\)](#)

For more details about the log Object and its methods, see [N/log Module](#).

util Object

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

util Object Members

- [util.isArray\(obj\)](#)
- [util.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()

Method Description	<p>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.</p> <p>Note: Consider this method a replacement for the <code>instanceOf</code> operator (which is not supported). SuiteScript 2.0 members are immutable; you cannot construct or modify a native SuiteScript 2.0 member. Consequently, if you attempt to call <code>instanceOf</code>, an undefined error is thrown.</p> <p>For information about <code>toString()</code> as a native JavaScript method, see https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/toString.</p>
Returns	The object type as a string
Supported Script Types	All script types
Governance	None
Since	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.MapContext"
```

...

JSON object

SuiteScript 2.0 supports the JavaScript Object Notation (JSON) standard. You can use the JSON object to parse text as a JSON object and convert strings to JSON notation. For more information, see [JSON.parse\(text\)](#) and [JSON.stringify\(obj\)](#).



Important: The following sections are included as a summary and are intended for reference only. For additional information about JSON, see <http://www.ietf.org/rfc/rfc4627.txt>.

JSON.parse(text)

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

Parameters

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
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)

Method Description	Converts a JavaScript object value to a key-value pair string in the JSON format. For more information about JSON object format, see http://www.ietf.org/rfc/rfc4627.txt .
Returns	Object
Supported Script Types	All script types
Governance	None
Global object	JSON object
Since	Version 2015 Release 2

Parameters

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

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

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
N/auth Module	Load the auth module when you want to change your NetSuite login credentials.
N/cache Module	Load the cache module to enable the caching of needed data and improve performance.
N/config Module	Load the config module when you want to access NetSuite configuration settings. See config.Type for a list of supported configuration pages.
N/crypto Module	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.
N/currency Module	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.
N/currentRecord Module	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.
N/email Module	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.
N/encode Module	Load the encode module when you want to convert a string to another type of encoding. See encode.Encoding 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/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.

Module	Description
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 log Object .
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/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"> ■ URL ■ Suitelet ■ Record ■ Task link ■ Saved search ■ Unsaved search
N/render Module	Load the render module to create forms or email from templates and to print to PDF or HTML.
N/runtime Module	Load the runtime module when you want to access the runtime settings for company, script, session, system, user, or version.
N/search Module	Load the search module to create and run ad-hoc or saved searches and analyze and iterate through the search results. You can search for a single record by keywords, create saved searches, search for duplicate records, or return a set of records that match filters you define.
N/sftp Module	Load the sftp module to connect to a remote FTP server via SFTP and transfer files.
N/sso Module	Load the sso module when you want to generate outbound single sign-on (SuiteSignOn) tokens
N/task Module	Load the task module to create tasks and place them in the internal NetSuite scheduling or task queue. Use the task module to schedule scripts, run Map/Reduce scripts, import CSV files, merge duplicate records, and execute asynchronous workflows.
N/transaction Module	Load the transaction module to void transactions.
N/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 util Object .

Module	Description
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/auth Module

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

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

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

N/auth Module Members

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

N/auth Module Script Sample

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

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

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

```

    });
}
changeEmailAndPassword();
});

```

auth.changeEmail(options)

Method Description	Method used to change the current user's NetSuite email address (user name).
Returns	<code>void</code>
Supported Script Types	Server-side scripts
Governance	10 usage units
Module	N/auth Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.password</code>	<code>string</code>	required	<ul style="list-style-type: none"> ■ The logged in user's NetSuite password. 	Version 2015 Release 2
<code>options.newEmail</code>	<code>string</code>	required	<ul style="list-style-type: none"> ■ The logged in user's NetSuite email address. 	Version 2015 Release 2
<code>options.onlyThisAccount</code>	<code>boolean true false</code>	optional	<ul style="list-style-type: none"> ■ 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>. 	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
<code>INVALID_PSWD</code>		The argument for <code>options.password</code> is invalid.
<code>INVALID_EMAIL</code>		The argument for <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: 'mypwd',
    newEmail: 'jwolf@netsuite.com',
    onlyThisAccount: true
});
...
//Add additional code
```

auth.changePassword(options)

Method Description	Method used to change the current user's NetSuite password.
Returns	void
Supported Script Types	Server-side scripts
Governance	10 usage units
Module	N/auth Module
Since	Version 2015 Release 2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.currentPassword	string	required	■ The logged in user's current NetSuite password.	Version 2015 Release 2
options.newPassword	string	required	■ The logged in user's new NetSuite password.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
INVALID_PSWD		The argument for options.currentPassword is invalid.

Error Code	Message	Thrown If
USER_ERROR		

Syntax



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

N/cache Module

Load the cache module to enable temporary, short-term storage of data. The cache module is supported by all server-side script types.

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

- [N/cache Module Members](#)
- [Cache Object Members](#)
- [N/cache Module Script Sample](#)

N/cache Module Members

Member Type	Name	Return Type / Value Type	Description
Object	cache.Cache	Object	Encapsulates a segment of memory that can be used to temporarily store data on a short-term basis.
Method	cache.getCache(options)	cache.Cache	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	cache.Scope	enum	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	Description
Method	Cache.get(options)	string	Retrieves a value from the cache based on a key that you provide. If the requested value is not present in the cache, the method calls the user-defined function identified by the method's option.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.
	Cache.put(options)	string	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.
	Cache.remove(options)	string	Removes a value from the cache.
Property	Cache.name	string	A label that identifies the cache.
	Cache.scope	string	A value that describes the availability of the cache. A cache can be made available to the current script only, to all scripts in the current bundle, or to all scripts in your NetSuite account.

N/cache Module Script Sample

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

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



Note: This sample depends on a CSV file that you would need to manually create before running the script.

```
/**
 * @NScriptType Suitelet
 * @NApiVersion 2.x
 */
define(['N/cache', '/SuiteScripts/zipCodes/ca/zipToCityIndexCacheLoader'],
    function (cache, lib){

    const ZIP_CODES_CACHE_NAME = 'ZIP_CODES_CACHE';
    const ZIP_TO_CITY_IDX_JSON = 'ZIP_TO_CITY_IDX_JSON';

    function getZipCodeToCityLookupObj(){
        var zipCache = cache.getCache({name: ZIP_CODES_CACHE_NAME});
        var zipCacheJson = zipCache.get({
            name: ZIP_TO_CITY_IDX_JSON
        });
        return zipCacheJson;
    }
});
```

```

        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.
get Time()) + ' ms');
    }
}
return {
    onRequest: onRequest
};
});

```

The following custom module returns the loader function used in the preceding code sample. The loader function shows how to use a CSV file to retrieve a value that was missing from a cache. This script does not need to include logic for placing the retrieved value into the cache — whenever a value is returned through the options.loader parameter, the value is automatically placed into the cache. For this reason, a loader function can serve as the sole method of populating a cache with values.

```

/**
 * @NApiVersion 2.0
 * @NModuleScope Public
 */
define(['N/file', 'N/cache'], function(file, cache){
    const ZIP_CODES_CSV_PATH = 'Resources/free-zipcode-ca-database-Primary.csv';
    function trimOuterQuotes(str){
        return (str || '').replace(/^\+/, '').replace(/\+$/, '');
    }

    function zipCodeDatabaseLoader(context){
        log.audit('Loading Zip Codes for ZIP_CODES_CACHE');
        var zipCodesCsvText = file.load({id: ZIP_CODES_CSV_PATH}).getContents();
        var zipToCityIndex = {};
        var csvLines = zipCodesCsvText.split('\n');
        util.each(csvLines.slice(1), function (el){
            var cells = el.split(',');
            var key = trimOuterQuotes(cells[0]);
            var value = trimOuterQuotes(cells[2]);
            if (parseInt(key, 10))
                zipToCityIndex[String(key)] = value;
        });
    }
});

```

```

    });
    return zipToCityIndex;
}

return {
    zipCodeDatabaseLoader : zipCodeDatabaseLoader
}

});

```

cache.Cache

Object Description	A segment of memory that can be used to temporarily store data needed by one script, by all scripts in a bundle, or by all scripts in the NetSuite account. For a complete list of this object's methods and properties, see Cache Object Members .
Supported Script Types	Server-side scripts
Module	N/cache Module
Since	2016.2

Cache.get(options)

Method Description	This method retrieves a string value from the cache. The value retrieved is identified by a key that you pass by using the options.key parameter. If a requested value is not present in the cache, the system calls the function identified by the options.loader parameter. This user-defined function should provide some logic for retrieving a value that is not in the cache.
Returns	String or null
Supported Script Types	Server-side scripts
Governance	1 unit if the value is present in the cache; 2 units if the loader function is used
Module	N/cache Module
Since	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	A user-defined function that returns the requested value if it is not already present in the cache. Additionally,

Parameter	Type	Required / Optional	Description
			<p>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.</p> <p>Note also that if the value returned by the loader is not a string, the system uses <code>JSON.stringify()</code> to convert the value before it is placed in the cache and returned. The maximum size of a value that can be placed in the cache is 500KB. When no loader is specified and a value is missing from the cache, the system returns null.</p>
<code>options.ttl</code>	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 before the <code>ttl</code> limit is reached.</p> <p>The default <code>ttl</code> value is no limit. The minimum value is 300 (five minutes).</p> <div style="border: 1px solid #f0e68c; padding: 10px; background-color: #fff;"> <p>! Important: 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.</p> </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 myValue = myCache.get({
    key: 'keyText',
    loader: loader,
    ttl: 18000
});
...
//Add additional code
```

Cache.put(options)

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

	Note: You can also place a value in a cache by using the <code>Cache.get(options)</code> method. In general, using the get method may result in a more efficient design.
Returns	Void
Supported Script Types	All server-side scripts
Governance	1 unit
Module	N/cache Module
Since	2016.2

Parameters

Note: The options parameter is a JavaScript object.			
Parameter	Type	Required / Optional	Description
<code>options.key</code>	string	required	An identifier of the value that is being cached. The maximum size of the cache key is 4 kilobytes.
<code>options.value</code>	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.

Syntax

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.put({
```

```

        key: 'keyText',
        value: 'valueText',
        ttl: 300
    });
...
//Add additional code

```

Cache.remove(options)

Method Description	Removes a value from the cache.
Returns	Void
Supported Script Types	All server-side scripts
Governance	1 unit
Module	N/cache Module
Since	2016.2

Parameters

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

Parameter	Type	Required / Optional	Description
options.key	string	required	An 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
...
myCache.remove({
    key: 'keyText'
});
...
//Add additional code

```

Cache.name

Property Description	A label that identifies a cache.
Type	string
Module	N/cache Module

Since	2016.2
-------	--------

Syntax



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

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

Cache.scope

Property Description	A label that describes the availability of the cache to other scripts.
Type	cache.Scope
Module	N/cache Module
Since	2016.2

Syntax



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

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

cache.getCache(options)

Method Description	Method used to create a new cache.Cache object.
Returns	cache.Cache
Supported Script Types	All server-side scripts
Governance	n/a

Module	N/cache Module
Since	2016.2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.name	string	required	A label that will identify the cache you are creating. The maximum size of the cache name is 1 kilobyte.
options.scope	string	optional, but if you do not set a value, the default of 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

Enum Description	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.
	Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Type	enum
Module	N/cache Module
Since	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"> ■ If the script is part of a bundle, the cache is available to all scripts in the same bundle. ■ If the script is not in a bundle, then the cache is available to all scripts not in a bundle.
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/config Module

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

See [config.Type](#) for a list of supported configuration objects.



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

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

N/config Module Members

Member Type	Name	Return Type / Value Type	Description
Method	config.load(options)	record.Record	Loads a record.Record object that encapsulates the specified configuration page.
Enum	config.Type	enum	Holds the string values for supported configuration objects. This enum is used to

Member Type	Name	Return Type / Value Type	Description
			set the value of the <i>Config.type</i> property.

N/config Module Script Sample

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

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

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)

Method Description	Method used to load a <code>record.Record</code> object that encapsulates the specified NetSuite configuration page.
--------------------	--

	After the configuration page loads, all preference names and IDs are available to get or set. For more information, see the help topic Preference Names and IDs .
Returns	<code>record.Record</code>
Supported Script Types	Server-side scripts
Governance	10 usage units
Module	N/config Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.type</code>	enum	required	<ul style="list-style-type: none"> ■ The NetSuite configuration page you want to access. See N/config Module for supported configuration pages. ■ Sets the value for the Record.type property. This property is read-only and cannot be changed after record.Record is loaded. ■ Use the config.Type enum to set the value. 	Version 2015 Release 2

Error Code	Message	Thrown If
<code>INVALID_RCRD_TYPE</code>		The <code>type</code> argument is invalid or 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/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

```

config.Type

Enum Description	Enumeration that holds the string values for supported configuration pages. This enum is used to set the value of the Record.type property. Note that the Record.type property is read-only.
Module	N/config Module
Since	Version 2015 Release 2

Values

Value	Configuration Page
USER_PREFERENCES	Set Preferences page (Home > Set Preferences) For more information about the fields on the page, see the help topic User Preferences .
COMPANY_INFORMATION	Company Information page (Setup > Company > Company Information) For more information about the fields on the page, see the help topic Company Information .
COMPANY_PREFERENCES	General Preferences page (Setup > Company > General Preferences) For more information about the fields on the page, see the help topic General Preferences .
ACCOUNTING_PREFERENCES	Accounting Preferences page (Setup > Accounting > Accounting Preferences) For more information about the fields on the page, see the help topic Accounting Preferences .
ACCOUNTING_PERIODS	Accounting Periods page (Setup > Accounting > Manage Accounting Periods) For more information about the fields on the page, see the help topic Accounting Periods .
TAX_PERIODS	Tax Periods page (Setup > Accounting > Manage Tax Periods) For more information about the fields on the page, see the help topic Tax Periods .
FEATURES	Enable Features page (Setup > Company > Enable Features) For more information about feature names and IDs, see the help topic Feature Names and IDs .

Syntax

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

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](#).

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

- [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	Description
Object	crypto.Cipher	Object	Encapsulates a cipher.
	crypto.CipherPayload	Object	Encapsulates a cipher payload.
	crypto.Decipher	Object	Encapsulates a decipher.
	crypto.Hash	Object	Encapsulates a hash.
	crypto.Hmac	Object	Encapsulates an hmac.
	crypto.SecretKey	Object	Encapsulates a secret key handle.

Member Type	Name	Return Type / Value Type	Description
Method	crypto.createCipher(options)	Object	Creates and returns a new crypto.Cipher Object.
	crypto.createDecipher(options)	Object	Creates and returns a new crypto.Decipher object.
	crypto.createHash(options)	Object	Creates and returns a new crypto.Hash Object.
	crypto.createHmac(options)	Object	Creates and returns a new crypto.Hmac Object.
	crypto.createSecretKey(options)	Object	Creates and returns a new crypto.SecretKey Object.
Enum	crypto.EncryptionAlg	string (read-only)	Holds the string values for supported encryption algorithms. Sets the options.algorithm parameter for crypto.createCipher(options) .
	crypto.HashAlg	string (read-only)	Holds the string values for supported hashing algorithms. Sets the value of the options.algorithm parameter for crypto.createHash(options) and crypto.createHmac(options) .
	crypto.Padding	string (read-only)	Holds the string values for supported cipher padding. Sets the options.padding parameter for crypto.createCipher(options) and crypto.createDecipher(options) .

Cipher Object Members

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

Member Type	Name	Return Type / Value Type	Description
Method	Cipher.update(options)	Object	Updates the clear data with the specified encoding
	Cipher.final(options)	Object	Returns the cipher data.

CipherPayload Object Members

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

Member Type	Name	Return Type / Value Type	Description
Property	CipherPayload.ciphertext	string	The result of the ciphering process.
	CipherPayload.iv	number	An initialization vector

Decipher Object Members

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

Member Type	Name	Return Type / Value Type	Description
Method	Decipher.final(options)	string	Returns the clear data.
	Decipher.update(options)	void	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	Description
Method	Hash.digest(options)	string	Calculates the digest of the data to be hashed.
	Hash.update(options)	void	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	Description
Method	Hmac.digest(options)	string	Gets the computed digest.
	Hmac.update(options)	void	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	Description
Property	Secretkey.guid	string	The GUID associated with the secret key.
	SecretKey.encoding	string	The encoding used for the clear text value of the secret key.

N/crypto Module Script Samples

Example 1

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

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

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

```
/**  
 * @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();  
        };  
    });
```

Example 2

Note: This sample script uses the `define` function. Note that you cannot use [Ad Hoc Debugging](#) to step though a `define` function. You must use [Deployed Debugging](#) to step through this script.

```
/**  
 * @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'  
                });  
                form.addSecretKeyField({  
                    id: 'mycredential',  
                    label: 'Credential',  
                });  
            }  
        }  
    });
```

```

        restrictToScriptIds: [runtime.getCurrentScript().id],
        restrictToCurrentUser: false
    }).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

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

Supported Script Types	Server-side scripts
Module	N/crypto Module
Since	Version 2015 Release 2

Syntax

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

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

Cipher.final(options)

Method Description	Method used to return the cipher data. Sets the output encoding for the crypto.CipherPayload object.
Returns	A crypto.CipherPayload Object
Supported Script Types	Server-side scripts
Governance	None
Module	N/crypto Module
Since	Version 2015 Release 2

Parameters

i Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.outputEncoding	enum	optional	The output encoding for a crypto.CipherPayload object. The default value is HEX. Use the encode.Encoding 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 Samples](#).

```
//Add additional code
```

```

...
crypto.createCipher({
    algorithm: crypto.EncryptionAlg.AES,
    key: sKey
});

var cipherPayload = cipher.final({
    outputEncoding: encode.Encoding.BASE_64
});
...
//Add additional code

```

Cipher.update(options)

Method Description	Method used to update the clear data with the specified encoding.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/crypto Module
Since	Version 2015 Release 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 <code>encode.Encoding</code> enum to set the value. 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 Samples](#).

```

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

```

crypto.CipherPayload

Object Description	Encapsulates a cipher payload. For a complete list of this object's methods and properties, see CipherPayload Object Members .
Supported Script Types	Server-side scripts
Module	N/crypto Module
Since	Version 2015 Release 2

Syntax

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

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

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

CipherPayload.ciphertext

Property Description	The result of the ciphering process. For example, to take the cipher payload and send it to another system.
Type	string
Module	N/crypto Module
Since	Version 2015 Release 2

Syntax

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

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

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

CipherPayload.iv

Property Description	Initialization vector for the cipher payload. You can pass in the iv value to crypto.createDecipher(options)
Type	string
Module	N/crypto Module
Since	Version 2015 Release 2

Syntax

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

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

crypto.Decipher

Object Description	Encapsulates a decipher. This object has methods that decrypt. For a complete list of this object's methods and properties, see Decipher Object Members .
Supported Script Types	Server-side scripts
Module	N/crypto Module
Since	Version 2015 Release 2

Syntax

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

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

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

Decipher.final(options)

Method Description	Method used to return the clear data.
Returns	string
Supported Script Types	Server-side scripts
Governance	None
Module	N/crypto Module
Since	Version 2015 Release 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 encode.Encoding 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 Samples](#).

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

Decipher.update(options)

Method Description	Method used to update cipher data with the specified encoding.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/crypto Module

Since

Version 2015 Release 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 encode.Encoding 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 Samples](#).

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

crypto.Hash

Object Description

Encapsulates a hash.

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

Supported Script Types

Server-side scripts

Module

[N/crypto Module](#)

Since

Version 2015 Release 2

Syntax



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

```
//Add additional code
...
var hashObj = crypto.createHash({
```

```

        algorithm: crypto.HashAlgorithm.SHA256
});
...
//Add additional code

```

Hash.digest(options)

Method Description	Calculates the digest of the data to be hashed.
Returns	A hash value as a string
Supported Script Types	Server-side scripts
Governance	None
Module	N/crypto Module
Since	Version 2015 Release 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 encode.Encoding enum. The default value is <code>HEX</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 Samples](#).

```

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

```

Hash.update(options)

Method Description	Method used to update clear data with the encoding specified.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/crypto Module

Since

Version 2015 Release 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 <code>encode.Encoding</code> 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 Samples](#).

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

crypto.Hmac

Object Description

Encapsulates an hmac.

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

Supported Script Types

Server-side scripts

Module

[N/crypto Module](#)

Since

Version 2015 Release 2

Syntax



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

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

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

Hmac.digest(options)

Method Description	Gets the computed digest.
Returns	An hmac value as a string
Supported Script Types	Server-side scripts
Governance	None
Module	N/crypto Module
Since	Version 2015 Release 2

Parameters

 **Note:** 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 <code>encode.Encoding</code> 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 Samples](#).

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

Hmac.update(options)

Method Description	Method used to update the clear data with the encoding specified.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None

Module	N/crypto Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
options.input	string	required	The hmac data to be updated.
options.inputEncoding	enum	optional	The input encoding. Set using the encode.Encoding 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 Samples](#).

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

crypto.SecretKey

Object Description	Encapsulates the handle to the key. The handler does not store the key value. It points to the key stored within the NetSuite system. The GUID is also required to find the key. For a complete list of this object's methods and properties, see SecretKey Object Members .
Supported Script Types	Server-side scripts
Module	N/crypto Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
```

```

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

//Add additional code

```

SecretKey.encoding

Property Description	The encoding used for the clear text value of the secret key.
Type	string
Module	N/crypto Module
Since	Version 2015 Release 2

Syntax



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

```

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

```

Secretkey.guid

Property Description	The GUID associated with the secret key.
Type	string
Module	N/crypto Module
Since	Version 2015 Release 2

Syntax



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

```

//Add additional code
...

```

```

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

```

crypto.createCipher(options)

Method Description	Method used to create and return a crypto.EncryptionAlg object.
	Note: The blockCipherMode is automatically set to CBC.
Returns	A crypto.EncryptionAlg object
Supported Script Types	Server-side scripts
Governance	None
Module	N/crypto Module
Since	Version 2015 Release 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 crypto.Hash enum.	Version 2015 Release 2
options.key	object	required	The crypto.SecretKey object.	Version 2015 Release 2
options.padding	string	optional	The padding for the cipher text. Set the value using the crypto.Padding enum. By default, the value is set to PKCS5Padding.	Version 2015 Release 2

Syntax

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

```

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

```

```

    key: sKey,
    padding: crypto.Padding.PKCS5Padding
});
...
//Add additional code

```

crypto.createDecipher(options)

Method Description	Method used to create a crypto.Decipher object.
	 Note: The blockCipherMode is automatically set to CBC.
Returns	A crypto.Decipher object.
Supported Script Types	Server-side scripts
Governance	None
Module	N/crypto Module
Since	Version 2015 Release 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 crypto.Hash enum.	Version 2015 Release 2
options.key	object	required	The crypto.SecretKey object used for encryption.	Version 2015 Release 2
options.padding	object	optional	The padding for the cipher. Set the value using the crypto.Padding enum.	Version 2015 Release 2
options.iv	string	required	The initialization vector that was used for encryption.	Version 2015 Release 2

Syntax

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

```

//Add additional code
...
var decipher = crypto.createDecipher({

```

```

        algorithm: crypto.EncryptionAlg.AES,
        key: sKey,
        padding: NoPadding,
        iv: '2311141720'
    });
...
//Add additional code

```

crypto.createHash(options)

Method Description	Method used to create a crypto.Hash object.
Returns	The crypto.Hash object created using this method.
Supported Script Types	Server-side scripts
Governance	None
Module	N/crypto Module
Since	Version 2015 Release 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 crypto.Hash enum.	Version 2015 Release 2

Syntax

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

```

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

```

crypto.createHmac(options)

Method Description	Method used to create a crypto.Hmac object.
---------------------------	---

Returns	A crypto.Hmac object.
Supported Script Types	Server-side scripts
Governance	None
Module	N/crypto Module
Since	Version 2015 Release 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 crypto.Hash enum to set this value.	Version 2015 Release 2
options.key	object	required	The crypto.SecretKey object.	Version 2015 Release 2

Syntax



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

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

crypto.createSecretKey(options)

Method Description	Method used to create a new crypto.SecretKey object. This method can take a GUID. Use Form.addCredentialField(options) to generate a value.
Returns	A crypto.SecretKey object
Supported Script Types	Server-side scripts
Governance	None
Module	N/crypto Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.guid	string	required	A GUID used to generate a secret key. The GUID can resolve to either data or metadata.	Version 2015 Release 2
options.encoding	enum	optional	Specifies the encoding for the SecureKey. Set this value using the encode.Encoding enum . The default value is HEX.	Version 2015 Release 2

Syntax

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

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

crypto.EncryptionAlg

Enum Description	Holds the string values for supported encryption algorithms. Sets the <code>options.algorithm</code> parameter for crypto.createCipher(options) .
	<p>i Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Module	N/crypto Module
Since	Version 2015 Release 2

Values

- AES

Syntax

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

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

crypto.HashAlg

Enum Description	Holds the string values for supported hashing algorithms. Sets the value of the <code>options.algorithm</code> parameter for <code>crypto.createHash(options)</code> and <code>crypto.createHmac(options)</code> .
Module	N/crypto Module
Since	Version 2015 Release 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 Samples](#).

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

crypto.Padding

Enum Description	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>Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Module	N/crypto Module
Since	Version 2015 Release 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 Samples](#).

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

N/currency Module

Load the N/currency module when you want to work with exchange rates within your NetSuite account. You can use this module to find the exchange rate between two currencies based on a certain date.

To use multiple currencies, the Multiple Currencies feature must be enabled. For information on enabling this feature, see the help topic [Enabling the Multiple Currencies Feature](#).

Note: Currency formatting is handled by the [N/format Module](#).

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

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

N/currency Module Member

Member Type	Name	Return Type / Value Type	Description
Method	currency.exchangeRate(options)	number	Returns an exchange rate between two currencies.

N/currency Module Script Sample

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

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

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

```

currency.exchangeRate(options)

Method Description	Method used to return the exchange rate between two currencies based on a certain date. The source currency is looked up relative to the target currency on the effective date. For example, if use British pounds for the source and US dollars for the target and the method returns '1.52', this means that if you were to enter an invoice today for a GBP customer in your USD subsidiary, the rate would be 1.52. The exchange rate values are sourced from the Currency Exchange Rate record. i Note: The Currency Exchange Rate record itself is not a scriptable record.
Returns	The exchange rate as a decimal number
Supported Script Types	Client and server-side scripts

Governance	10 units
Module	N/currency Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.date	Date	optional	<ul style="list-style-type: none"> ■ Pass in a new Date object. For example, <code>date: new Date('7/28/2015')</code> ■ If <code>date</code> is not specified, then it defaults to today (the current date). ■ The date determines the exchange rate in effect. If there are multiple rates, it is the latest entry on that date. ■ Use the same date format as your NetSuite account. 	Version 2015 Release 2
options.source	number string	required	<ul style="list-style-type: none"> ■ The internal ID or three-letter ISO code for the currency you are converting from. ■ For example, you can use either 1 (internal ID) or USD (currency code). ■ If the Multiple Currencies feature is enabled, from your account, you can view a list of all the currency internal IDs and ISO codes at Lists > Accounting > Currencies. 	Version 2015 Release 2
options.target	number string	required	<ul style="list-style-type: none"> ■ The internal ID or three-letter ISO code for the currency you are converting to. 	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
MISSING_REQD_ARGUMENT	exchangeRate: Missing a required argument: <source/target>	The source or target 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 source or target argument is invalid. If the Multiple Currencies feature is enabled, from your account, you can view a list of currency internal IDs and ISO codes at Lists > 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

You use the currentRecord module to access the record that is active in the current client-side context. You can use this module in both of the following types of scripts:

- **Entry point client scripts** — These scripts use the `@NScriptType ClientScript` annotation (see [SuiteScript 2.0 JSDoc Validation](#)). For these scripts, the currentRecord module is automatically loaded when the script is executed. Additionally, a `currentRecord.CurrentRecord` object is automatically created for the script. This object can be accessed by the `scriptContext` object that is passed through each of the [Client Script Entry Points](#). For an example, see [Client Script Sample](#).
- **Client-side custom modules** — These scripts do not use an `@NScriptType` annotation (see [SuiteScript 2.0 Custom Modules](#)). For these scripts, you must manually load the currentRecord module by naming it in the script's define statement. Additionally, you must actively retrieve a `currentRecord.CurrentRecord` object by using the `currentRecord.get()` or `currentRecord.get.promise()` method. For an example, see [N/currentRecord Module Script Sample](#).

Like the [N/record Module](#), the currentRecord module provides access to body and sublist fields. However, the record module is recommended for server scripts and for cases where a client-side script needs to interact with a record other than the currently active record. By contrast, the currentRecord module is recommended for client-side scripts that need to interact with the currently active record.

Additionally, the functionality of the two modules varies slightly. For example, the currentRecord module does not permit the editing of subrecords, although subrecords can be retrieved in view mode. For additional details, see the following topics:

- [N/currentRecord Module Members](#)

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

N/currentRecord Module Members

Member Type	Name	Return Type / Value Type	Description
Object	currentRecord. CurrentRecord	Object	Represents the record active on the current page.
	currentRecord.Field	Object	Represents a body or sublist field.
Method	currentRecord.get()	currentRecord. CurrentRecord	Retrieves a record object that represents the current record.
	currentRecord.get. promise()	Promise	Retrieves a promise for a record object that represents the current record.

CurrentRecord Object Members

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

Member Type	Name	Return Type / Value Type	Description
Method	CurrentRecord.cancelLine(options)	currentRecord. CurrentRecord	Cancels the changes made to the currently selected line.
	CurrentRecord.commitLine(options)	currentRecord. CurrentRecord	Commits the currently selected line.
	CurrentRecord.findMatrix SublistLineWithValue(options)	number	Returns the line number of the first line that contains the specified value in the matrix column.
	CurrentRecord.findSublistL ineWithValue(options)	number	Gets the line number for the first occurrence of a field value in a sublist.
	CurrentRecord.getCurrent MatrixSublistValue(options)	number Date string array boolean true false	Gets the value for the currently selected line in the matrix.
	CurrentRecord.getCurrent SublistIndex(options)	number	Gets the line number of the currently selected line.
	CurrentRecord.getCurrent SublistSubrecord(options)	currentRecord. CurrentRecord	Gets the subrecord for the associated sublist field on the current line. The subrecord object is retrieved in view mode.

Member Type	Name	Return Type / Value Type	Description
	CurrentRecord.getCurrentSublistText(options)	number Date string array boolean true false	Gets the value of the field in the currently selected line by text representation.
	CurrentRecord.getCurrentSublistValue(options)	number Date string array boolean true false	Gets the value of the field in the currently selected line.
	CurrentRecord.getField(options)	record.Field	Gets a field object from the record.
	CurrentRecord.getLineCount(options)	number	Returns the number of lines in the sublist.
	CurrentRecord.getMatrixHeaderCount(options)	number	Returns the number of columns for the specified matrix.
	CurrentRecord.getMatrixHeaderField(options)	record.Field	Gets the field for the specified header in the matrix.
	CurrentRecord.getMatrixHeaderValue(options)	number Date string array boolean true false	Gets the value for the associated header in the matrix.
	CurrentRecord.getMatrixSublistField(options)	record.Field	Gets the field for the specified sublist in the matrix.
	CurrentRecord.getMatrixSublistValue(options)	number Date string array boolean true false	Gets the value for the associated field in the matrix.
	CurrentRecord.getSublist(options)	record.Sublist	Gets the specified sublist object.
	CurrentRecord.getSublistField(options)	record.Field	Gets the specified field object from the sublist.
	CurrentRecord.getSublistText(options)	string	Gets the value of the field in a sublist by a string representation.
	CurrentRecord.getSublistValue(options)	number Date string array boolean true false	Gets the value of the field in a sublist.
	CurrentRecord.getSubrecord(options)	currentRecord. CurrentRecord	Gets the subrecord associated with the field. The subrecord object is retrieved in view mode.
	CurrentRecord.getText(options)	string	Gets the value of the field by a string representation.
	CurrentRecord.getValue(options)	number Date string array boolean true false	Gets the value of the field.

Member Type	Name	Return Type / Value Type	Description
	CurrentRecord.hasCurrentSublistSubrecord(options)	boolean true false	Returns a value indicating whether the associated sublist field has a subrecord on the current line.
	CurrentRecord.hasSublistSubrecord(options)	boolean true false	Returns a value indicating whether the associated sublist field contains a subrecord.
	CurrentRecord.hasSubrecord(options)	boolean true false	Indicates whether the field has a subrecord.
	CurrentRecord.insertLine(options)	currentRecord. CurrentRecord	Inserts a new line in a sublist.
	CurrentRecord.removeCurrentSublistSubrecord(options)	currentRecord. CurrentRecord	Removes the subrecord for the associated sublist field on the current line.
	CurrentRecord.removeLine(options)	currentRecord. CurrentRecord	Removes a line from a sublist.
	CurrentRecord.removeSubrecord(options)	currentRecord. CurrentRecord	Removes the subrecord associated with the field.
	CurrentRecord.selectLine(options)	void	Selects a line item in a sublist.
	CurrentRecord.selectNewLine(options)	currentRecord. CurrentRecord	Selects a new line at the end of the sublist.
	CurrentRecord.setCurrentMatrixSublistValue(options)	currentRecord. CurrentRecord	Sets the value for the currently selected line in the matrix.
	CurrentRecord.setCurrentSublistText(options)	currentRecord. CurrentRecord	Sets the value of the field in the currently selected line using a string representation.
	CurrentRecord.setCurrentSublistValue(options)	currentRecord. CurrentRecord	Sets the value of the field in the currently selected line.
	CurrentRecord.setMatrixHeaderValue(options)	currentRecord. CurrentRecord	Sets the value for the associated header in the matrix.
	CurrentRecord.setMatrixSublistValue(options)	currentRecord. CurrentRecord	Sets the value for the associated field in the matrix.
	CurrentRecord.setText(options)	currentRecord. CurrentRecord	Sets the value of the field using a string representation.
	CurrentRecord.setValue(options)	currentRecord. CurrentRecord	Sets the value of the field.

Member Type	Name	Return Type / Value Type	Description
Property	CurrentRecord.id	number (read-only)	Returns the internal record ID.
	CurrentRecord.isDynamic	boolean true false (read-only)	Indicates whether the record is dynamic.
	CurrentRecord.type	string (read-only)	Returns the record type.

Field Object Members

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

Member Type	Name	Return Type / Value Type	Description
Method	Field.getSelectOptions(options)	array	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	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	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)	Returns the internal ID of a standard or custom body or sublist field.
	Field.isDisabled	boolean true false	Returns true if the standard or custom field is disabled on the record form, or false otherwise.
	Field.isDisplay	boolean true false	Returns true if the field is set to display on the record form, or false otherwise. This property is read-only for sublist fields.
	Field.isMandatory	boolean true false	Returns true if the standard or custom field is mandatory on the record form, or false otherwise.

Member Type	Name	Return Type / Value Type	Description
	Field.isPopup	boolean true false (read-only)	Returns true if the field is a popup list field, or false otherwise.
	Field.isReadOnly	boolean true false	Returns true if the field on the record form cannot be edited, or false otherwise. For textarea fields, this property can be read or written to. For all other fields, this property is read-only.
	Field.isVisible	boolean true false (read-only)	Returns true if the field is visible on the record form, or false otherwise.
	Field.label	string (read-only)	Returns the UI label for a standard or custom field body or sublist field.
	Field.sublistId	string (read-only)	Returns the ID of the sublist associated with the specified sublist field.
	Field.type	string (read-only)	Returns the type of a body or sublist field.

N/currentRecord Module Script Sample

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

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

```
/**
 * @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,
                fireSlavingSync: true
            });
            var actValue = record.getValue({
                fieldId: 'custpage_textfield'
            });
            record.setValue({
                fieldId: 'custpage_resultfield',
                value: actValue
            });
        }
    }
});
```

```
        value: actValue,
        ignoreFieldChange: true,
        fireSlavingSync: true
    });
},
test_set_getCurrentSublistValue: function() {
    var record = currentRecord.get();
    record.setCurrentSublistValue({
        sublistId: 'itemvendor',
        fieldId: 'custpage_subtextfield',
        value: 'Sublist Value',
        ignoreFieldChange: true,
        fireSlavingSync: true
    });
    var actValue = record.getCurrentSublistValue({
        sublistId: 'itemvendor',
        fieldId: 'custpage_subtextfield'
    });
    record.setValue({
        fieldId: 'custpage_sublist_resultfield',
        value: actValue,
        ignoreFieldChange: true,
        fireSlavingSync: true
    });
},
},
});
```

The following example is a user event script deployed on a non-inventory item record. Before the record loads, the script updates the form used by the record to add new text fields, a sublist, and buttons that call the clientDemo.js methods. The buttons access the current record and set values for some of the form's fields.

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

```

currentRecord.CurrentRecord

Object Description	Encapsulates the record active on the current page.
Supported Script Types	Client scripts
Module	N/currentRecord Module
Since	2016.2

Syntax



Important: The following code snippets show the syntax for this member. These snippets are not a functional examples. For a complete script example, see [N/currentRecord Module Script Sample](#) and [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)

Method Description	Cancels the currently selected line on a sublist.
Returns	The <code>currentRecord.CurrentRecord</code> object that called the method.
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	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.cancelLine({
    sublistId: 'item'
});
...
//Add additional code
```

CurrentRecord.commitLine(options)

Method Description	Commits the currently selected line on a sublist.
Returns	The currentRecord.CurrentRecord object that called the method.
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	2016.2

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 Using the SuiteScript Records Browser .	2016.2

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/currentRecord Module Script Sample](#).

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

CurrentRecord.findMatrixSublistLineWithValue(options)

Method Description	Returns the line number of the first instance where a specified value is found in a specified column of the matrix.
Returns	number
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	2016.2

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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The ID of the matrix field.	2016.2
options.value	number	required	The value to search for.	2016.2

Parameter	Type	Required / Optional	Description	Since
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)

Method Description	Returns the line number for the first occurrence of a field value in a sublist.
Returns	A line number as a number, or -1 if not found.
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	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)

Method Description	Gets the value for the currently selected line in the matrix.
Returns	number Date string array boolean true false
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of the matrix field.	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)

Method Description	Returns the line number of the currently selected line.
Returns	number
Supported Script Types	Client scripts

Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	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)

Method Description	Gets the subrecord for the associated sublist field on the current line. The subrecord object is retrieved in view mode.
Returns	currentRecord.CurrentRecord
Supported Script Types	Client scripts

Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	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 objSubrecord = objRecord.getCurrentSublistSubrecord({
    sublistId: 'item',
    fieldId: 'item'
});
...
//Add additional code
```

CurrentRecord.getCurrentSublistText(options)

Method Description	Returns a text representation of the field value in the currently selected line.
Returns	string
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of a standard or custom sublist 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 fieldName = objRecord.getCurrentSublistText({
    sublistId: 'item',
    fieldId: 'item'
});
...
//Add additional code
```

CurrentRecord.getCurrentSublistValue(options)

Method Description	Returns the value of a sublist field on the currently selected sublist line.
Returns	number Date string array boolean true false
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of a standard or custom sublist 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 sublistValue = objRecord.getCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'item'
});
...
//Add additional code
```

CurrentRecord.getField(options)

Method Description	Returns a field object from a record.
Returns	currentRecord.Field
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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

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)

Method Description	Returns the number of lines in a sublist.
Returns	number
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	

Syntax

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

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

CurrentRecord.getMatrixHeaderCount(options)

Method Description	Returns the number of columns for the specified matrix.
Returns	number
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of 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 numLines = objRecord.getMatrixHeaderCount({
    sublistId: 'item',
    fieldId: 'item'
});
...
//Add additional code
```

CurrentRecord.getMatrixHeaderField(options)

Method Description	Gets the field for the specified header in the matrix.
Returns	record.Field
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	2016.2

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 that contains the matrix. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of the matrix field.	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)

Method Description	Gets the value for the associated header in the matrix.
Returns	number Date string array boolean true false
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of the matrix field.	2016.2

Parameter	Type	Required / Optional	Description	Since
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 value = objRecord.getMatrixHeaderValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 12
});
...
//Add additional code
```

CurrentRecord.getMatrixSublistField(options)

Method Description	Gets the field for the specified sublist in the matrix.
Returns	record.Field
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	2016.2

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 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 Using the SuiteScript Records Browser .	
options.fieldId	string	required	The internal ID of the matrix field.	2016.2
options.column	number	required	The column number for the field.	2016.2
options.line	number	required	The line 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.getMatrixSublistField({
    sublistId: 'item',
    fieldId: 'item',
    column: 12,
    line: 3
});
...
//Add additional code
```

CurrentRecord.getMatrixSublistValue(options)

Method Description	Gets the value for the associated field in the matrix.
Returns	number Date string array boolean true false
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of the matrix field.	2016.2
options.column	number	required	The column number for the field.	2016.2
options.line	number	required	The line 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 value = objRecord.getMatrixSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 12,
    line: 3
});
...
//Add additional code
```

CurrentRecord.getSublist(options)

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

Returns	record.Sublist
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	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)

Method Description	Returns a field object from a sublist.
Returns	record.Field
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	2016.2
options.line	number	required	The line number for 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 objField = objRecord.getSublistField({
    sublistId: 'item',
    fieldId: 'item',
    line: 3
});
...
//Add additional code
```

CurrentRecord.getSublistText(options)

Method Description	Returns the value of a sublist field in a text representation.
Returns	string

Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	2016.2
options.line	number	required	The line number for 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 sublistFieldName = objRecord.getSublistText({
    sublistId: 'item',
    fieldId: 'item',
    line: 3
});
...
//Add additional code
```

CurrentRecord.getSublistValue(options)

Method Description	Returns the value of a sublist field.
Returns	number Date string array boolean true false
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	2016.2
options.line	number	required	The line number for 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 sublistFieldValue = objRecord.getSublistValue({
```

```

    sublistId: 'item',
    fieldId: 'item',
    line: 3
});
...
//Add additional code

```

CurrentRecord.getSubrecord(options)

Method Description	Gets the subrecord associated with the field. The subrecord object is available in view mode.
Returns	currentRecord.CurrentRecord
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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

Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
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/currentRecord Module Script Sample](#).

```
//Add additional code
```

```

...
var sublistFieldValue = objRecord.getSubrecord({
    fieldId: 'subrecord'
});
...
//Add additional code

```

CurrentRecord.getText(options)

Method Description	Returns the text representation of a field value.
Returns	string
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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

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)

Method Description	Returns the value of a field.
Returns	number Date string array boolean true false
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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

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)

Method Description	Returns a value indicating whether the associated sublist field has a sublist on the current line. This method can only be used on dynamic records.
---------------------------	--

Returns	boolean true false
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of a subrecord.	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)

Method Description	Returns a value indicating whether the associated sublist field contains a subrecord.
Returns	boolean true false
Supported Script Types	Client scripts
Governance	None

Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of a subrecord.	2016.2
options.line	number	required	The line number for the field.	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)

Method Description	Returns a value indicating whether the field contains a subrecord.
Returns	boolean true false
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module

Since	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 field that may contain a subrecord.	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)

Method Description	Inserts a sublist line.
Returns	currentRecord.CurrentRecord
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	
options.line	number	required	The line number to insert.	2016.2
options.ignoreRecalc	boolean true false	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.insertLine({
    sublistId: 'item',
    line: 3,
    ignoreRecalc: true
});
...
//Add additional code
```

CurrentRecord.removeCurrentSublistSubrecord(options)

Method Description	Removes the subrecord for the associated sublist field.
Returns	<code>currentRecord.CurrentRecord</code>
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	2016.2

Parameters

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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	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)

Method Description	Removes a sublist line.
Returns	currentRecord.CurrentRecord
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	2016.2

Parameters

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 Using the SuiteScript Records Browser .	
options.line	number	required	The line number of the sublist to remove.	2016.2
options.ignoreRecalc	boolean true false	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)

Method Description	Removes the subrecord for the associated field.
Returns	<code>currentRecord.CurrentRecord</code>
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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.	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)

Method Description	Selects an existing line in a sublist.
Returns	currentRecord.CurrentRecord
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	2016.2

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 Using the SuiteScript Records Browser .	2016.2

Parameter	Type	Required / Optional	Description	Since
options.line	number	required	The line number to select in the sublist.	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 lineNum = objRecord.selectLine({
    sublistId: 'item',
    line: 3
});
...
//Add additional code
```

CurrentRecord.selectNewLine(options)

Method Description	Selects a new line at the end of a sublist.
Returns	<code>currentRecord.CurrentRecord</code>
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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	2016.2

Parameter	Type	Required / Optional	Description	Since
			information, see the help topic Using the SuiteScript Records Browser .	

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)

Method Description	Sets the value for the line currently selected in the matrix. This method is not available for standard records.
Returns	currentRecord.CurrentRecord
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	2016.2

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 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 Using the SuiteScript Records Browser .	
options.fieldId	string	required	The internal ID of the matrix field.	2016.2
options.column	number	required	The column number for the field.	2016.2
options.value	number Date string array boolean true false	required	<p>The value to set the field to. The value type must correspond to the field type being set. For example:</p> <ul style="list-style-type: none"> ■ Text, Radio and Select fields accept string values. ■ Checkbox fields accept Boolean values. ■ Date and DateTime fields accept Date values. ■ Integer, Float, Currency and Percent fields accept number values. 	2016.2
options.ignoreFieldChange	boolean true false	optional	If set to true, the field change and slaving event is ignored. By default, this value is false.	2016.2
options.fireSlavingSync	boolean true false	optional	Indicates whether to perform slaving synchronously. By default, this value is false.	2016.2

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/currentRecord Module Script Sample](#).

```
//Add additional code
...
objRecord.setCurrentMatrixSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 3,
    value: false,
    ignoreFieldChange: true,
    fireSlavingSync: true
});
...
//Add additional code
```

CurrentRecord.setCurrentSublistText(options)

Method Description	Sets the value for the field in the currently selected line by a text representation.
Returns	currentRecord.CurrentRecord
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	2016.2
options.text	string	required	The text to set the value to.	2016.2

Parameter	Type	Required / Optional	Description	Since
options.ignoreFieldChange	boolean true false	optional	If set to true, the field change and slaving event is ignored. By default, this value is false.	2016.2
options.fireSlavingSync	boolean true false	optional	Indicates whether to perform slaving synchronously. By default, this value is false.	2016.2

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

 **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,
    fireSlavingSync: true
});
...
//Add additional code
```

CurrentRecord.setCurrentSublistValue(options)

Method Description	Sets the value for the field in the currently selected line.
Returns	currentRecord.CurrentRecord
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module

Since	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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	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"> ■ Text, Radio and Select fields accept string values. ■ Checkbox fields accept Boolean values. ■ Date and DateTime fields accept Date values. ■ Integer, Float, Currency and Percent fields accept number values. 	2016.2
options.ignoreFieldChange	boolean true false	optional	If set to <code>true</code> , the field change and slaving event is ignored. By default, this value is <code>false</code> .	2016.2
options.fireSlavingSync	boolean true false	optional	Indicates whether to perform slaving synchronously. By default, this value is <code>false</code> .	2016.2

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/currentRecord Module Script Sample](#).

```
//Add additional code
...
objRecord.setCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    value: true,
    ignoreFieldChange: true
});
...
//Add additional code
```

CurrentRecord.setMatrixHeaderValue(options)

Method Description	Sets the value for the associated header in the matrix.
Returns	currentRecord.CurrentRecord
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	2016.2

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 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 Using the SuiteScript Records Browser .	
options.fieldId	string	required	The internal ID of the matrix field.	2016.2
options.column	number	required	The column number for the field.	2016.2
options.value	number Date string array boolean true false	required	<p>The value to set the field to. The value type must correspond to the field type being set. For example:</p> <ul style="list-style-type: none"> ■ Text, Radio and Select fields accept string values. ■ Checkbox fields accept Boolean values. ■ Date and DateTime fields accept Date values. ■ Integer, Float, Currency and Percent fields accept number values. 	2016.2
options.ignoreFieldChange	boolean true false	optional	If set to <code>true</code> , the field change and slaving event is ignored. By default, this value is <code>false</code> .	2016.2
options.fireSlavingSync	boolean true false	optional	Indicates whether to perform slaving synchronously. By default, this value is <code>false</code> .	2016.2

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/currentRecord Module Script Sample](#).

```
//Add additional code
...
objRecord.setMatrixHeaderValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 3,
    value: false,
    ignoreFieldChange: true,
    fireSlavingSync: true
});
...
//Add additional code
```

CurrentRecord.setMatrixSublistValue(options)

Method Description	Sets the value for the associated field in the matrix.
Returns	currentRecord.CurrentRecord
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	2016.2

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 that contains the matrix. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The internal ID of the matrix field.	2016.2
options.column	number	required	The column number for the field.	2016.2

Parameter	Type	Required / Optional	Description	Since
options.line	number	required	The line number for the field.	2016.2
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"> ■ Text, Radio and Select fields accept string values. ■ Checkbox fields accept Boolean values. ■ Date and DateTime fields accept Date values. ■ Integer, Float, Currency and Percent fields accept number values. 	2016.2

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.setMatrixSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 12,
    line: 3,
    value: true
});
...
//Add additional code
```

CurrentRecord.setText(options)

Method Description	Sets the value of the field by a text representation.
Returns	currentRecord.CurrentRecord
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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
options.text	string	required	The text to change the field value to.	2016.2
options.ignoreFieldChange	boolean true false	optional	If set to true, the field change and slaving event is ignored. By default, this value is false.	2016.2
options.fireSlavingSync	boolean true false	optional	Indicates whether to perform slaving synchronously. By default, this value is false.	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
...
objRecord.setText({
    fieldId: 'item',
    text: 'value',
    ignoreFieldChange: true,
```

```

    fireSlavingSync: true
});
...
//Add additional code

```

CurrentRecord.setValue(options)

Method Description	Sets the value of a field.
Returns	currentRecord.CurrentRecord
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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
options.value	number Date string array boolean true false	required	<p>The value to set the field to. The value type must correspond to the field type being set. For example:</p> <ul style="list-style-type: none"> ■ Text, Radio, Select and Multi-Select fields accept string values. ■ Checkbox fields accept Boolean values. ■ Date and DateTime fields accept Date values. ■ Integer, Float, Currency and Percent fields accept number values. 	2016.2
options.ignoreFieldChange	boolean true false	optional	If set to true, the field change and	2016.2

Parameter	Type	Required / Optional	Description	Since
			slaving event is ignored. By default, this value is false.	
options.fireSlavingSync	boolean true false	optional	Indicates whether to perform slaving synchronously. By default, this value is false.	2016.2

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/currentRecord Module Script Sample](#).

```
//Add additional code
...
objRecord.setValue({
    fieldId: 'item',
    value: true,
    ignoreFieldChange: true,
    fireSlavingSync: true
});
...
//Add additional code
```

CurrentRecord.id

Property Description	The internal ID of a specific record.
Type	number (read-only)
Module	N/currentRecord Module
Since	2016.2

Syntax

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

```
//Add additional code
```

```

...
var recordid = record.id;
...
//Add additional code

```

CurrentRecord.isDynamic

Property Description	Indicates whether the record is in dynamic mode. This value is set when the record is created or accessed.
Type	boolean true false (read-only)
Module	N/currentRecord Module
Since	2016.2

Syntax

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

```

//Add additional code
...
if (record.isDynamic) {
    ...
}
...
//Add additional code

```

CurrentRecord.type

Property Description	The record type. This value is set with the record.Type enum during record creation.
Type	string (read-only)
Module	N/currentRecord Module
Since	2016.2

Syntax

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

```

//Add additional code
...
var recordtype = record.type;
...
//Add additional code

```

currentRecord.Field

Object Description	Encapsulates a body or sublist field on the current record. Use the following methods to access the Field object:
	<ul style="list-style-type: none"> ▪ CurrentRecord.getField(options) ▪ CurrentRecord.getSublistField(options) <p>For a complete list of this object's methods and properties, see N/currentRecord Module.</p>
Supported Script Types	Client scripts
Module	N/currentRecord Module
Since	2016.2

Syntax

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

```
//Add additional code
...
var currentRecordField = currentRecord.getField({
    fieldId: 'entity'
});
...
//Add additional code
}
```

Field.getSelectOptions(options)

Method Description	Returns an array of available options on a standard or custom select, multiselect, or radio field as key-value pairs. Only the first 1,000 available options are returned. This function returns an array in the following format: <code>[{value: 5, text: 'abc'},{value: 6, text: '123'}]</code> This function returns Type Error if the field is not a supported field for this method.
Returns	array
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module

Since	Version 2016 Release 2
-------	------------------------

Parameters

 Note: 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.  Note: Filter values are case insensitive.	Version 2016 Release 2
options.operator	string	Required	The following operators are supported: <ul style="list-style-type: none">■ contains (default)■ is■ startswith	Version 2016 Release 2

Syntax

 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 options = objField.getSelectOptions({
    filter : 'C',
    operator : 'startswith'
});
...
//Add additional code
```

Field.insertSelectOption(options)

Method Description	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 custpage .
Returns	Void
Supported Script Types	Client
Governance	None
Module	N/currentRecord Module

Since	Version 2016 Release 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.	Version 2016 Release 2
options.text	string	Required	The label that represents the option in the UI.	Version 2016 Release 2
options.isSelected	boolean	Optional	Determines whether the option is selected by default. If not specified, this value defaults to false.	Version 2016 Release 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 custpage .

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_selectfield'
});

// Insert a new option.

field.insertSelectOption({
    value: 'Option1',
    text: 'alpha'
});
...
```

```
//Add additional code
```

Field.removeSelectOption(options)

Method Description	Removes a select option from certain types of select and multiselect fields. This method is usable only in select fields that were added by a front-end Suitelet or beforeLoad user event script. The IDs for these fields always have a prefix of custpage .
Returns	Void
Supported Script Types	Client
Governance	None
Module	N/currentRecord Module
Since	Version 2016 Release 2

Parameters

i 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: ... field.removeSelectOption({ value: null, }); ...	Version 2016 Release 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 custpage .

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

Property Description	Returns the internal ID of a standard or custom body or sublist field.
Type	string (read-only)
Module	N/currentRecord Module
Since	Version 2016 Release 2

Syntax



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

Property Description	Returns the UI label for a standard or custom field body or sublist field.
Type	string (read-only)
Module	N/currentRecord Module
Since	Version 2016 Release 2

Syntax

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

Property Description	Returns <code>true</code> if the standard or custom field is mandatory on the record form, or <code>false</code> otherwise.
Type	<code>boolean true false</code>
Module	N/currentRecord Module
Since	Version 2016 Release 2

Syntax

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

Property Description	Returns <code>true</code> if the standard or custom field is disabled on the record form, or <code>false</code> otherwise.
Type	<code>boolean true false</code>
Module	N/currentRecord Module
Since	Version 2016 Release 2

Syntax

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

Property Description	Returns <code>true</code> if the field is a popup list field, or <code>false</code> otherwise.
Type	<code>boolean true false</code> (read-only)
Module	N/currentRecord Module
Since	Version 2016 Release 2

Syntax

 **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

Property Description	Returns <code>true</code> if the field is set to display on the record form, or <code>false</code> otherwise. Fields can be a part of a record even if they are not displayed on the record form. This property is read-only for sublist fields.
Type	<code>boolean true false</code>
Module	N/currentRecord Module
Since	Version 2016 Release 2

Syntax

 **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

Property Description	Returns <code>true</code> if the field is visible on the record form, or <code>false</code> otherwise.
Type	<code>boolean true false (read-only)</code>
Module	N/currentRecord Module
Since	Version 2016 Release 2

Syntax

 **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

Property Description	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.
Type	<code>boolean true false</code>
Module	N/currentRecord Module
Since	Version 2016 Release 2

Syntax

 **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

Property Description	Returns the sublist ID for the specified sublist field.
Type	string (read-only)
Module	N/currentRecord Module
Since	Version 2016 Release 2

Syntax

 **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

Property Description	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.
Type	string (read-only)
Module	N/currentRecord Module
Since	Version 2016 Release 2

Syntax

 **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.get()

Method Description	Retrieves a currentRecord object that represents the record active on the current page.
Returns	<code>currentRecord.CurrentRecord</code>

Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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()

Method Description	Retrieves a promise for a currentRecord object that represents the record active on the current page.
Returns	Promise
Supported Script Types	Client scripts
Governance	None
Module	N/currentRecord Module
Since	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 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.promise();
...
//Add additional code
```

N/email Module

Load the N/email module when you want to send email messages from within NetSuite. You can use the N/email module to send regular, bulk, and campaign email.

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

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

N/email Module Members

Member Type	Name	Return Type / Value Type	Description
Method	email.send(options)	void	Sends email to an individual or group of recipients and receives bounceback notifications.
	email.send.promise(options)	void	Sends email asynchronously to an individual or group of recipients and receives bounceback notifications.
	email.sendBulk(options)	void	Sends bulk email.
	email.sendBulk.promise(options)	void	Sends bulk email asynchronously.
	email.sendCampaignEvent(options)	number	Sends lead nurturing campaigns (drip marketing email).
	email.sendCampaignEvent.promise(options)	number	Sends lead nurturing campaigns (drip marketing email) asynchronously.

N/email Module Script Sample

The following example sends email with an attachment.

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

```
/**
 *@NApiVersion 2.x
 */
require(['N/email', 'N/record', 'N/file'],
    function(email, record, file) {
```

```

function sendEmailWithAttachement() {
    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
        }
    });
}
sendEmailWithAttachement();
});

```

email.send(options)

Method Description	Method used to send transactional email and receive bounceback notifications if the email is not successfully delivered. A maximum of 10 recipients (recipient + cc + bcc) is allowed. The total message size (including attachments) must be 15MB or less. The size of individual attachments cannot exceed 10BM.
Returns	void
Supported Script Types	Client and server-side scripts
Governance	20 usage units
Module	N/email Module
Since	Version 2015 Release 2

Parameters

i Note: The options parameter is a JavaScript object.				
Parameter	Type	Required / Optional	Description	Since
options.author	number	required	<ul style="list-style-type: none"> ■ Internal ID of the email sender. ■ To find the internal ID of the sender in the UI, go to Lists > Employees. 	Version 2015 Release 2
options.recipients	number[] string[]	required	<ul style="list-style-type: none"> ■ 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. <p>i Note: Only the first recipient displays on the Communication tab (under the Recipient column). To view all recipients, click View to open the Message record.</p>	Version 2015 Release 2
options.replyTo	string	optional	<ul style="list-style-type: none"> ■ The email address that appears in the reply-to header when an email is sent out. ■ You can use either a single external email address or a generic email address created by the Email Capture Plug-in. 	Version 2015 Release 2
options.cc	number[] string[]	optional	<ul style="list-style-type: none"> ■ 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. 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> ■ A maximum of 10 recipients (recipient + cc + bcc) is allowed. 	
options.bcc	number[] string[]	optional	<ul style="list-style-type: none"> ■ The internal ID or email address of the recipient to blind copy. ■ For multiple recipients, use an array of internal IDs or email addresses. You can use an array that contains a combination of internal IDs and email addresses. ■ A maximum of 10 recipients (recipient + cc + bcc) is allowed. 	Version 2015 Release 2
options.subject	string	required	<ul style="list-style-type: none"> ■ Subject of the outgoing message. 	Version 2015 Release 2
options.body	string	required	<ul style="list-style-type: none"> ■ Contents of the email ■ SuiteScript formats the body of the email in either plain text or HTML. If HTML tags are present, the message is formatted as HTML. Otherwise, the message is formatted in plain text. To display XML as plain text, use an HTML<pre> tag around XML content. 	Version 2015 Release 2
options.attachments	file.File	optional	<ul style="list-style-type: none"> ■ 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 15MB or less. <p>Note: Supported for server-side scripts only.</p>	Version 2015 Release 2
options.relatedRecords	Object	optional	<ul style="list-style-type: none"> ■ Object that contains key/value pairs to associate the Message record with related records (including custom records). ■ See the relatedRecords table for more information 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.isInternalOnly	boolean true false	required	<ul style="list-style-type: none"> ■ If true, the Message record is not visible to an external Entity (for example, a customer or contact). ■ The default value is false. 	Version 2015 Release 2



Note: The relatedRecords parameter is a JavaScript object and the table below lists its properties.

relatedRecords

You can associate the sent email with an array of internal records using key/value pairs.

Parameter	Type	Required / Optional	Description	Since
transactionId	number	optional	The Transaction record(s) associated with the Message record. Use for transaction and opportunity record types.	Version 2015 Release 2
activityId	number	optional	The Activity record(s) attached to the Email Message record. Use for Case and Campaign record types.	Version 2015 Release 2
entityId	number	optional	The Entity record(s) attached to the Email Message record. Use for all Entity record types (for example, customer, contact)	Version 2015 Release 2
customRecord	Object	optional	The custom record(s) attached to the Email Message record. For custom records you must specify both the record ID and the record type ID.	Version 2015 Release 2
customRecord.id	number	optional	The instance ID for the custom record attached to the Email Message record <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Note: If you use this parameter, customRecord.recordType is required. </div>	Version 2015 Release 2
customRecord.recordType	string	optional	The custom record type attached to the Message record	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<p>Note: If you use this parameter, <code>customRecord.id</code> is required.</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/email Module Script Sample](#).

```
//Add additional code
...
var senderId = -5;
var recipientEmail = 'notify@myCompany.com';
var timeStamp = new Date().getUTCMilliseconds();
var recipient = record.create({
    type: record.Type.CUSTOMER,
    isDynamic: true
});
recipient.setValue({
    fieldId: 'subsidiary',
    value: '1'
});
recipient.setValue({
    fieldId: 'companyname',
    value: 'Test Company' + timeStamp
});
recipient.setValue({
    fieldId: 'email',
    value: recipientEmail
});
var recipientId = recipient.save();
var fileObj = file.load({
    id: 88
});
email.send({
    author: senderId,
    recipients: recipientId,
    subject: 'Test Sample Email Module',
    body: 'email body',
    attachments: [fileObj],
    relatedRecords: {
        entityId: recipientId
    }
});
...
//Add additional code
```

email.send.promise(options)

Method Description	Method used to send transactional email asynchronously and receive bounceback notifications if the email is not successfully delivered.
	<p>Note: For information about the parameters and errors thrown for this method, see email.send(options). For additional information about promises, see Promise object.</p>
Returns	void
Synchronous Version	email.send(options)
Supported Script Types	All client-side scripts
Governance	20 usage units
Module	N/email Module
Since	Version 2015 Release 2

email.sendBulk(options)

Method Description	This method is used to send bulk email when a bounceback notification is not required.
	<p>Note: This API normally uses a bulk email server to send messages. If you need to increase the successful delivery rate of an email, use email.send(options) so that a transactional email server is used.</p>
Returns	void
Supported Script Types	Client and server-side scripts
Governance	10 usage units
Module	N/email Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.author	number	required	<ul style="list-style-type: none"> ■ Internal ID of the email sender. ■ To find the internal ID of the sender in the UI, go to Lists > Employees. 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.recipients	number[] string[]	required	<ul style="list-style-type: none"> ■ 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. <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p>Note: Only the first recipient displays on the Communication tab (under the Recipient column). To view all recipients, click View to open the Message record.</p> </div>	Version 2015 Release 2
options.replyTo	string	optional	<ul style="list-style-type: none"> ■ The email address that appears in the reply-to header when an email is sent out. ■ You can use either a single external email address or a generic email address created by the Email Capture Plug-in. 	Version 2015 Release 2
options.cc	number[] string[]	optional	<ul style="list-style-type: none"> ■ 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. 	Version 2015 Release 2
options.bcc	number[] string[]	optional	<ul style="list-style-type: none"> ■ The internal ID or email address of the recipient to blind copy. ■ For multiple recipients, use an array of internal IDs or email addresses. 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<p>You can use an array that contains a combination of internal IDs and email addresses.</p> <ul style="list-style-type: none"> ■ A maximum of 10 recipients (recipient + cc + bcc) is allowed. 	
options.subject	string	required	<ul style="list-style-type: none"> ■ Subject of the outgoing message. 	Version 2015 Release 2
options.body	string	required	<ul style="list-style-type: none"> ■ Contents of the email ■ SuiteScript formats the body of the email in either plain text or HTML. If HTML tags are present, the message is formatted as HTML. Otherwise, the message is formatted in plain text. To display XML as plain text, use an HTML<pre> tag around XML content. 	Version 2015 Release 2
options.attachments	file.File	optional	<ul style="list-style-type: none"> ■ 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 15MB or less. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Note: Supported for server-side scripts only. </div>	Version 2015 Release 2
options.relatedRecords	Object	optional	<ul style="list-style-type: none"> ■ Object that contains key/value pairs to associate the Message record with related records (including custom records). ■ See the relatedRecords table for more information 	Version 2015 Release 2
options.isInternalOnly	boolean true false	optional	<ul style="list-style-type: none"> ■ If true, the Message record is not visible to an external Entity (for example, a customer or contact). ■ The default value is false. 	Version 2015 Release 2



Note: The relatedRecords parameter is a JavaScript object and the table below lists its properties.

relatedRecords

You can associate the sent email with an array of internal records using key/value pairs.

Parameter	Type	Required / Optional	Description	Since
transactionId	number	optional	<ul style="list-style-type: none"> ■ The Transaction record(s) attached to the Message record ■ Use for transaction and opportunity record types. 	Version 2015 Release 2
activityId	number	optional	<ul style="list-style-type: none"> ■ The Activity record(s) attached to the Email Message record ■ Use for Case and Campaign record types. 	Version 2015 Release 2
entityId	number	optional	<ul style="list-style-type: none"> ■ The Entity record(s) attached to the Email Message record ■ Use for all Entity record types (for example, customer, contact) 	Version 2015 Release 2
customRecord	Object	optional	<ul style="list-style-type: none"> ■ The custom record(s) attached to the Email Message record ■ For custom records you must specify both the record ID and the record type ID. 	Version 2015 Release 2
customRecord.id	number	optional	<ul style="list-style-type: none"> ■ The instance ID for the custom record attached to the Email Message record 	Version 2015 Release 2
customRecord.recordType	string	optional	<ul style="list-style-type: none"> ■ The custom record type attached to the Message record 	Version 2015 Release 2

Syntax

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)

Method Description	This method is used to send bulk email asynchronously when a bounceback notification is not required. Note: For information about the parameters and errors thrown for this method, see email.sendBulk(options) . For additional information about promises, see Promise object .
Returns	void
Synchronous Version	email.sendBulk(options)
Supported Script Types	All client-side scripts
Governance	10 usage units
Module	N/email Module
Since	Version 2015 Release 2

email.sendCampaignEvent(options)

Method Description	Method used to send a single “on-demand” campaign email to a specified recipient and return a campaign response ID to track the email. Email (campaignemail) sublists are not supported. The campaign must use a Lead Nurturing (campaigndrip) sublist.
---------------------------	---

	Note: This API normally uses a bulk email server to send messages. If you need to increase the successful delivery rate of an email, use <code>email.send(options)</code> so that a transactional email server is used.
Returns	A campaign response ID (tracking code) as number If the email fails to send, the value returned is -1.
Supported Script Types	Client and server-side scripts
Governance	10 usage units
Module	N/email Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
<code>options.campaignEventId</code>	number	required	<ul style="list-style-type: none"> ■ The internal ID of the campaign event. <p>Note: The campaign must use a Lead Nurturing (campaigndrip) sublist.</p>	Version 2015 Release 2
<code>options.recipientId</code>	number	required	<ul style="list-style-type: none"> ■ The internal ID of the recipient. <p>Note: The recipient's record must contain an email address.</p>	Version 2015 Release 2

Syntax

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
...
email.sendCampaignEvent({
    campaignEventId: -8,
    recipientId: 142,
```

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

email.sendCampaignEvent.promise(options)

Method Description	Method used to send a single “on-demand” campaign email asynchronously to a specified recipient and return a campaign response ID to track the email.
	<p>Note: For information about the parameters and errors thrown for this method, see email.sendCampaignEvent(options). For additional information about promises, see Promise object.</p>
Returns	A campaign response ID (tracking code) as a number. If the email fails to send, the value returned is -1.
Synchronous Version	email.sendCampaignEvent(options)
Supported Script Types	All client-side scripts
Governance	10 usage units
Module	N/email Module
Since	Version 2015 Release 2

N/encode Module

This module exposes string encoding and decoding functionality. Load the N/encode module when you want to convert a string to another type of encoding.

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

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

N/encode Module Members

Member Type	Name	Return Type / Value Type	Description
Method	encode.convert(options)	string	Converts a string to another type of encoding and returns the re-encoded string.
Enum	encode.Encoding	enum	Holds the string values for supported encoding specifications.

N/encode Module Script Sample

The following example converts a string to a different encoding.

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

```
/**
 * @NApiVersion 2.x
 */
require(['N/encode'],
    function(encode) {
        function convertStringToDifferentEncoding() {
            var stringInput = "T@ B@st Stri@ Bg 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)

Method Description	Converts a string to another type of encoding.
Returns	The re-encoded string
Supported Script Types	Server-side scripts
Governance	None
Module	N/encode Module
Since	Version 2015 Release 1

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.string	string	required	The string to encode.	Version 2015 Release 1
options.inputEncoding	string	required	The encoding used on the input string. The default value is <code>UTF_8</code> .	Version 2015 Release 1

Parameter	Type	Required / Optional	Description	Since
			Use the <code>encode.Encoding</code> to set the value.	
<code>options.outputEncoding</code>	string	required	The encoding to apply to the output string. The default value is <code>UTF_8</code> . Use the <code>encode.Encoding</code> to set the value.	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/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

Enum Description	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 N/crypto Module or N/encode Module . Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Module	N/encode Module
Since	Version 2015 Release 1

Values

- `UTF_8`
- `BASE_16`
- `BASE_32`
- `BASE_64`

- BASE_64_URL_SAFE
- HEX

Syntax



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

```
//Add additional code
...
var reencoded = encode.convert({
    string: LOREM_IPS,
    inputEncoding: encode.Encoding.BASE_64,
    outputEncoding: encode.Encoding.UTF_8
});
...
//Add additional code
```

N/error Module

Load the error module when you want to create your own custom SuiteScript errors. Use these custom errors in try-catch statements to abort script execution.

- [N/error Module Members](#)
- [SuiteScriptError Object Members](#)
- [UserEventError Object Members](#)
- [N/error Module Script Sample](#)

N/error Module Members

Member Type	Name	Return Type / Value Type	Description
Object	error.SuiteScriptError	Object	Encapsulates a SuiteScript error thrown by any script type that is not a user event script.
	error.UserEventError	Object	Encapsulates a SuiteScript error thrown by a user event script.
Method	error.create(options)	error.SuiteScriptError or error.UserEventError	Creates a new error.SuiteScriptError or error.UserEventError object.

SuiteScriptError Object Members

The following members are called on `error.SuiteScriptError`.

Member Type	Name	Return Type / Value Type	Description
Property	SuiteScriptError.name	string (read-only)	User-defined error code.
	SuiteScriptError.message	string (read-only)	Text that displays on the SuiteScript Execution Log, in the Details column.
	SuiteScriptError.id	string (read-only)	Error ID that is automatically generated when a new error is created.
	SuiteScriptError.stack	Array of strings (read-only)	A list of method calls that the script is executing when the error is thrown.

UserEventError Object Members

The following members are called on `error.UserEventError`.

Member Type	Name	Return Type / Value Type	Description
Property	UserEventError.name	string (read-only)	User-defined error code.
	UserEventError.message	string (read-only)	Text that displays on the SuiteScript Execution Log, in the Details column.
	UserEventError.eventType	string (read-only)	User event type (beforeLoad, beforeSubmit, afterSubmit)
	UserEventError.id	string (read-only)	Error ID that is automatically generated when a new error is created.
	UserEventError.recordId	string (read-only)	Internal ID of the submitted record that triggered the script. This property only holds a value when the error is thrown by an afterSubmit user event script.
	UserEventError.stack	Array of strings (read-only)	A list of method calls that the script is executing when the error is thrown.

N/error Module Script Sample

The following example creates an error.

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

```
/**  
 * @NApiVersion 2.x  
 */  
require(['N/error'],
```

```

function(error) {
    function createError() {
        var errorObj = error.create({
            name: 'MY_CODE',
            message: 'my error details',
            notifyOff: true
        });
    }
    createError();
}
);

```

error.SuiteScriptError

Object Description	Encapsulates a SuiteScript error for any script type that is not a user event script. Use this object in a try-catch statement to abort script execution. Create a new custom error (<code>error.SuiteScriptError</code>) with the <code>file.create(options)</code> method. The <code>file.create(options)</code> method returns <code>error.SuiteScriptError</code> when it is called in any server-side script that is not a user event script.
<p> Note: When <code>file.create(options)</code> is called in a user event script, it returns <code>error.UserEventError</code>.</p>	
Supported Script Types	All server-side scripts that are not user event scripts.
Module	N/error Module
Since	Version 2015 Release 2

Syntax



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 errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
...
//Add additional code

```

SuiteScriptError.id

Property Description	Error ID that is automatically generated when a new error is created.
-----------------------------	---

Type	This property is read-only. string
Module	N/file Module
Since	Version 2015 Release 2

Syntax

⚠️ 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 errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error ID: " + errorObj.id);
...
//Add additional code
```

SuiteScriptError.message

Property Description	Text that displays on the SuiteScript Execution Log, in the Details column. This property is read-only.
Type	string
Module	N/file Module
Since	Version 2015 Release 2

Syntax

⚠️ 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 errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Message: " + errorObj.message);
...
//Add additional code
```

SuiteScriptError.name

Property Description	A user-defined name (error code). This property is read-only.
Type	string
Module	N/file Module
Since	Version 2015 Release 2

Syntax

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 errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Code: " + errorObj.name);
...
//Add additional code
```

SuiteScriptError.stack

Property Description	A list of method calls that the script is executing when the error is thrown. The most recently executed method is listed at the top. This property is read-only.
Type	Array of strings
Module	N/file Module
Since	Version 2015 Release 2

Syntax

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 errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Stack: " + errorObj.stack);
...
```

```
//Add additional code
```

error.UserEventError

Object Description	Encapsulates a SuiteScript error for user event scripts. Use this object in a try-catch statement to abort script execution. Create a new custom error (<code>error.UserEventError</code>) with the <code>file.create(options)</code> method The <code>file.create(options)</code> method returns <code>error.UserEventError</code> when it is called in a user event script.
<p>Note: When <code>file.create(options)</code> is called in a server-side script that is not a user event script, it returns <code>error.SuiteScriptError</code>.</p>	
Supported Script Types	User event scripts
Module	N/error Module
Since	Version 2015 Release 2

Syntax



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 errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
...
//Add additional code
```

UserEventError.eventType

Property Description	The user event type. Holds one of the following values: <ul style="list-style-type: none"> ▪ beforeLoad ▪ beforeSubmit ▪ afterSubmit This property is read-only.
Type	string
Module	N/file Module
Since	Version 2015 Release 2

Syntax



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 errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("User Event Type: " + errorObj.eventType);
...
//Add additional code
```

UserEventError.stack

Property Description	A list of method calls that the script is executing when the error is thrown. The most recently executed method is listed at the top. This property is read-only.
Type	Array of strings
Module	N/file Module
Since	Version 2015 Release 2

Syntax



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 errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Stack: " + errorObj.stack);
...
//Add additional code
```

UserEventError.id

Property Description	Error ID that is automatically generated when a new error is created. This property is read-only.
Type	string
Module	N/file Module

SinceVersion 2015 Release 2

Syntax

 **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 errorObj = error.create({
  name: 'MY_CODE',
  message: 'my error details',
  notifyOff: false
});
log.debug("Error ID: " + errorObj.id);
...
//Add additional code
```

UserEventError.message

Property Description	Text that displays on the SuiteScript Execution Log, in the Details column. This property is read-only.
Type	string
Module	N/file Module
Since	Version 2015 Release 2

Syntax

 **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 errorObj = error.create({
  name: 'MY_CODE',
  message: 'my error details',
  notifyOff: false
});
log.debug("Error Message: " + errorObj.message);
...
//Add additional code
```

UserEventError.name

Property Description	A user-defined name (error code).
-----------------------------	-----------------------------------

Type	This property is read-only. string
Module	N/file Module
Since	Version 2015 Release 2

Syntax

⚠ 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 errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Code: " + errorObj.name);
...
//Add additional code
```

UserEventError.recordId

Property Description	The internal ID of the submitted record that triggered the script. This property only holds a value when the error is thrown by an afterSubmit user event script. This property is read-only.
Type	string
Module	N/file Module
Since	Version 2015 Release 2

Syntax

⚠ 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 errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Submitted Record ID: " + errorObj.recordId);
...
//Add additional code
```

error.create(options)

Method Description	Method used to create a new <code>error.SuiteScriptError</code> or <code>error.UserEventError</code> object. Use this custom error in a try-catch statement to abort script execution.
Returns	One of the following: <ul style="list-style-type: none"> ■ An <code>error.UserEventError</code> object if the script throwing the error is a user event script. ■ An <code>error.SuiteScriptError</code> object if the script throwing the error is any other server-side script.
Supported Script Types	Server-side scripts
Governance	None
Module	N/error Module
Since	Version 2015 Release 2

Parameters

i Note: 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.name	string	required	<ul style="list-style-type: none"> ■ A user-defined name (error code). ■ Sets the value for the property <code>file.load(options)</code>. 	Version 2015 Release 2
options.message	string	required	<ul style="list-style-type: none"> ■ The error message displayed. This value displays on the Execution Log, in the Details column. ■ The default value is null. ■ Sets the value for the property <code>File.description</code>. <pre>//test var errorObj = error.create({ name: 'MY_CODE', message: 'my error details', notifyOff: false }); log.debug("Error Code: " + errorObj.name);</pre>	Version 2015 Release 2
options.notifyOff	boolean true false	optional	<ul style="list-style-type: none"> ■ Sets whether email notification is suppressed. 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> ■ The default value is false. ■ If set to false, when this error is thrown, the system emails the users identified on the applicable script record's Unhandled Errors subtab. For additional information on the Unhandled Errors subtab, see Creating a Script Record. 	

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
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Code: " + errorObj.name);
...
//Add additional code
```

N/file Module

Load the file module when you want to work with files within NetSuite. You can use this module to upload files to the NetSuite file cabinet. You can also use this module to send files as attachments without uploading them to the file cabinet.

Methods that load content in memory, such as [File.getContents\(\)](#), have a 10MB size limit. This limit does not apply when content is streamed, such as when [File.save\(\)](#) is called.

- N/file Module Members
- File Object Members
- N/file Module Script Sample

N/file Module Members

Member Type	Name	Return Type / Value Type	Description
Object	file.File	object	Encapsulates a file within NetSuite.
Method	file.create(options)	file.File	Creates a new file.File .
	file.delete(options)	void	Deletes an existing file.File from the NetSuite file cabinet.
	file.load(options)	file.File	Loads an existing file.File from the NetSuite file cabinet.
Enum	file.Encoding	enum	Sets the value of the File.encoding property.
	file.Type	enum	Sets the value of the File.fileType property.

File Object Members

The following members are called on [file.File](#).

Member Type	Name	Return Type / Value Type	Description
Method	File.appendLine(options)	file.File Object	Inserts a line to the end of a CSV or text file.
	File.getContents()	string	Returns the content of a file in string format.
	File.lines.iterator()	boolean true false	Calls a developer-defined function for each line. Returns false when line processing stops.
	File.resetStream()	void	Resets the file stream to its previous state.
	File.save()	number	Saves a new or updated file to the file cabinet.
Property	File.description	string	Description of a file.
	File.encoding	string	Character encoding on a file.
	File.fileType	enum	File type of a file.
	File.folder	number	Internal ID of the folder that houses a file within the NetSuite file cabinet.
	File.id	number (read-only)	Internal ID of a file in the NetSuite file cabinet.
	File.isInactive	boolean true false	Inactive status of a file. If set to true, the file is inactive.

Member Type	Name	Return Type / Value Type	Description
	File.isOnline	boolean true false	"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)	Indicates whether a file type is text-based.
	File.name	string	Name of a file.
	File.path	string (read-only)	Relative path to a file in the NetSuite file cabinet.
	File.size	number (read-only)	Size of a file in bytes.
	File.url	string (read-only)	URL of a file.

N/file Module Script Sample

i Note: These sample scripts use the `require` function so that you can copy into the debugger and test. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see [SuiteScript 2.0 – Script Architecture](#) and [SuiteScript 2.0 Script Types and Entry Points](#).

Example 1

The following example creates and saves a file to the file cabinet.

```
/** 
 * @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();
    });

```

Example 2

The following sample creates and saves a file to the file cabinet. It also sets the values of `File.isOnline` and the `File.folder` properties.

```
/**
```

```
*@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();
  }
);

```

Example 3

```
require(['N/file', 'N/error', 'N/log'],
  function (file, error, log)
  {

    // In this sample we will compute the total for the
    // second column value in a csv file.
    //
    // date,amount
    // 10/21/14,200.0
    // 10/21/15,210.2
    // 10/21/16,250.3

    // Create the CSV file
    var csvFile = file.create({
      name: 'data.csv',
      contents: 'date,amount\n',
      folder: 39,
      fileType: 'CSV'
    });
    csvFile.appendLine({
      value: '10/21/14,200.0'
    });
    csvFile.appendLine({
      value: '10/21/15,210.2'
    });
    csvFile.appendLine({
      value: '10/21/16,250.3'
    });
    var csv fileId = csvFile.save();
  }
);

```

```

// This variable will store the total.
var total = 0.0;

// Load the file and
// process all the lines
var invoiceFile = file.load({
    id: csvFileDialog
});
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: ' + lineValue
        });
    total += lineAmount;
    return true;
});

// By the time you are here, the total variable is
// set to 660.5
log.debug({
    title: 'total',
    details: total
});
}
)

```

file.File

Object Description	Encapsulates a file within NetSuite.
--------------------	--------------------------------------



Note: This object only encapsulates a file's metadata. Content is only loaded into memory (and returned as a string) when you call the `File.getContents()`. Content from CSV or text files can be accessed line by line using `File.appendLine(options)` or `File.lines.iterator()`.



Important: Binary content must be base64 encoded.

Create a new `file.File` Object (up to 10MB in size) with the `file.create(options)` method.

After you create a new `file.File`, you can:

- upload it to the NetSuite file cabinet with the `File.save()` method.
- attach it to an email or fax without saving it to the file cabinet.



Important: If you want to save the file to the NetSuite file cabinet, you must set a NetSuite file cabinet folder with the `File.folder` property. You must do this before you call `File.save()`.

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

Supported Script Types	Server-side scripts
Module	N/file Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
var fileObj = file.create({
    name: 'test.txt',
    fileType: file.Type.PLAINTEXT,
    contents: 'Hello World\nHello World'
});
fileObj.folder = 30;
var fileId = fileObj.save();
...
//Add additional code
```

File.getContents()

Method Description	Method used to return the content of the file.
	<p> Important: Content held in memory is limited to 10MB.</p>
	<p> Note: You can access CSV or text files (including files over 10MB) using <code>File.appendLine(options)</code> or <code>File.lines.iterator()</code>.</p>
Returns	The file content as a string
Supported Script Types	Server-side scripts
Governance	None

Module	N/file Module
Since	Version 2015 Release 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 [N/file Module Script Sample](#).

```
//Add additional code
...
var fileObj = file.load({
    id: 145
});
if (fileObj.size < 10485760){
    fileObj.getContents();
}
...
//Add additional code
```

File.appendLine(options)

Method Description	Method used to insert a line to the end of a file. This method can be used on text or .csv files.
Returns	file.File Object
Supported Script Types	Server-side scripts
Governance	None
Module	N/file Module
Since	Version 2017 Release 1

Parameters

 **Note:** 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.	Version 2017 Release 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 <code>File.appendLine(options)</code> after calling <code>File.lines.iterator()</code> . To avoid receiving the error, 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 [N/file Module Script Sample](#).

```
//Add additional code
...
var fileObj = file.load({
    id: 145
});
fileObj.appendLine({
    value: 'hello world'
});
...
//Add additional code
```

File.lines.iterator()

Method Description	Method used to pass the next line as an argument to a developer-defined function. You can call this method multiple times to loop over the file contents as a stream. Return <code>false</code> to stop the loop. Return true to continue the loop. By default, <code>false</code> is returned when the end of the file is reached. This method can be used on text or .csv files.
---------------------------	--

	 Important: Content held in memory is limited to 10MB. Therefore, each line must be less than 10MB.
Returns	Boolean true false
Supported Script Types	Server-side scripts
Governance	None
Module	N/file Module
Since	Version 2017 Release 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.	Version 2017 Release 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 [N/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: ' + lineValue
    });
total += lineAmount;
return true;
});
...
//Add additional code

```

File.resetStream()

Method Description	Method used to reset the file contents. Serves as an undo action on any unsaved content written with File.appendLine(options) or File.lines.iterator() . Use this method to reset the reading and writing streams that may have been opened by File.appendLine(options) or File.lines.iterator() . The line pointer (or read iterator) is also set to its previous state. This method can be used on text or .csv files.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/file Module
Since	Version 2017 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/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()

Method Description	<p>Method used to:</p> <ul style="list-style-type: none"> ■ Upload a new file to the NetSuite file cabinet. ■ Save an updated file to the NetSuite file cabinet. <p>Note: 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>Important: 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>
Returns	The internal ID of the file as a number.
Supported Script Types	Server-side scripts
Governance	20 usage units
Module	N/file Module
Since	Version 2015 Release 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 [N/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

Property Description	The description of a file. In the UI, the value of <code>description</code> displays in the Description field on the file record.
Type	string
Module	N/file Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
var fileObj = file.load({
    id: 'Images/myImageFile.jpg'
});
fileObj.description = 'my test file';
var fileId = fileObj.save();
...
//Add additional code
```

File.encoding

Property Description	The character encoding on a file. Value is set with the <code>file.Encoding</code> enum.
Type	string
Module	N/file Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
var fileObj = file.create({
    name      : 'test.txt',
    fileType: file.Type.PLAINTEXT,
    contents: 'Hello World\nHello World'
```

```

    });
fileObj.encoding = file.Encoding.MAC_ROMAN;
fileObj.folder = 30;
var fileId = fileObj.save();
...
//Add additional code

```

File.fileType

Property Description	The file type of a file. This property is read-only. You must set the file type by passing in a file.Type enum value to file.create(options) .
Type	enum
Module	N/file Module
Since	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property after it is set with file.create(options) .

Syntax



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

```

//Add additional code
...
var fileObj = file.load({
    id: 145
});
log.debug({
    details: "File Type: " + fileObj.fileType
});
...
//Add additional code

```

File.folder

Property Description	The internal ID of a file's folder within the NetSuite file cabinet. Before you upload a file to the NetSuite file cabinet with File.save() , you must set its file cabinet folder with the <code>folder</code> property.
-----------------------------	--

Type	number string
Module	N/file Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var fileObj = file.create({
    name: 'test.txt',
    fileType: file.Type.PLAINTEXT,
    contents: 'Hello World\nHello World'
});
fileObj.folder = 30;
var fileId = fileObj.save();
...
//Add additional code
```

File.id

Property Description	The internal ID of the file within the NetSuite file cabinet. This value is automatically generated by NetSuite. This property is read-only.
Type	number
Module	N/file Module
Since	Version 2015 Release 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/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

Property Description	The inactive status of a file. If set to true, the file is inactive. The default value is <code>false</code> . When a file is inactive, it does not display in the UI unless you select Show Inactives on the File Cabinet page.
Type	<code>boolean true false</code>
Module	N/file Module
Since	Version 2015 Release 2

Syntax



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

```

//Add additional code  
...  
var fileObj = file.load({  
    id: 'Images/myImageFile.jpg'  
});  
fileObj.name = 'myOldImageFile.jpg';  
fileObj.isInactive = true;  
var fileId = fileObj.save();  
...  
//Add additional code

```

File.isOnline

Property Description	The Available without Login status of a file. If set to <code>true</code> , users can download the file outside of a current NetSuite login session. The default value is <code>false</code> .
	<p>Important: This property holds the value of the Available without Login setting found on the file record. It does not reflect the value of the Available Without Login setting found on the Suitelet script deployment record.</p> <p>The Available without Login setting is primarily used for SuiteCommerce web sites. When this setting is enabled, web sites can access media files in the NetSuite file cabinet without a current NetSuite login session.</p>

Type	boolean true false
Module	N/file Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var fileObj = file.load({
    id: 'Images/myImageFile.jpg'
});
fileObj.isOnline = true;
var fileId = fileObj.save();
...
//Add additional code
```

File.isText

Property Description	Indicates whether a file type is text-based. This property is read-only.
Type	boolean true false
Module	N/file Module
Since	Version 2015 Release 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/file Module Script Sample](#).

```
//Add additional code
...
var fileObj = file.load({
    id: 145
});
if (fileObj.isText === true){
    ...
}
```

```

}
...
//Add additional code

```

File.name

Property Description	The name of a file.
Type	string
Module	N/file Module
Since	Version 2015 Release 2

Syntax

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

```

//Add additional code
...
var fileObj = file.load({
    id: 'Images/myImageFile.jpg'
});
fileObj.name = 'myOldImageFile.jpg';
var fileId = fileObj.save();
...
//Add additional code

```

File.path

Property Description	<p>The relative path to a file in the NetSuite file cabinet.</p> <p>Note: If the folder is not set with the <code>file.create(options)</code> method, this property holds the file name until the <code>File.folder</code> property is defined.</p>
	This property is read-only.
Type	string
Module	N/file Module
Since	Version 2015 Release 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/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

Property Description	The size of a file in bytes. This property is read-only.
	Note: You can use this value to determine if the file is within size limits for File.getContents() . Size will reflect any lines you have streamed into a file. For example, the original file size plus lines appended.
Type	number
Module	N/file Module
Since	Version 2015 Release 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/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

Property Description	The URL of a file. This property is read-only.
Type	string
Module	N/file Module
Since	Version 2015 Release 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/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)

Method Description	Method used to create a new file in the NetSuite file cabinet.
	Important: Content held in memory is limited to 10MB.
Returns	file.File
Supported Script Types	Server-side scripts
Governance	None
Module	N/file Module
Since	Version 2015 Release 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"> ■ The file name and extension. ■ Sets the value for the File.name property. 	Version 2015 Release 2
options.fileType	enum	required	<ul style="list-style-type: none"> ■ The file type. ■ Sets the value for the File.fileType property. This property is read-only and cannot be changed after the file is created. ■ Use the file.Type enum to set the value. 	Version 2015 Release 2
options.contents	string	optional	<ul style="list-style-type: none"> ■ The file content. ■ File content is lazy loaded; there is no property for it. ■ If the file type is binary (for example, PDF), the file content must be base64 encoded. 	Version 2015 Release 2
options.description	string	optional	<ul style="list-style-type: none"> ■ The file description. In the UI, the value of description displays the Description field on the file record. ■ Sets the value for the File.description property. 	2016.2
options.folder	number	optional	<ul style="list-style-type: none"> ■ The internal ID of the folder within the NetSuite file cabinet. You must set the file cabinet folder before you upload a file to the NetSuite file cabinet with File.save(). ■ Sets the value for the File.folder property. 	Version 2016 Release 1
options.encoding	string	optional	<ul style="list-style-type: none"> ■ The character encoding on a file. 	2016.2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> ■ Sets the value for the File.encoding property. ■ Use the file.Encoding enum to set the value. 	
options.isInactive	boolean false true	optional	<ul style="list-style-type: none"> ■ The inactive status of a file. If set to true, the file is inactive. The default value is false. When a file is inactive, it does not display in the UI unless you select Show Inactives on the File Cabinet page. ■ Sets the value for the File.isInactive property. 	2016.2
options.isOnline	boolean false true	optional	<ul style="list-style-type: none"> ■ The Available without Login status of a file. If set to true, users can download the file outside of a current netSuite login session. The default value is false. ■ Sets the value for the File.isOnline property. 	2016.2

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 File.fileType 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 [N/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)

Method Description	Method used to delete an existing file from the NetSuite file cabinet.
Returns	The internal ID of the deleted file
Supported Script Types	Server-side scripts
Governance	20 usage units
Module	N/file Module
Since	Version 2015 Release 2

Parameters

i Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.id	number string	required	<ul style="list-style-type: none"> ■ Internal ID of the file. ■ To find the internal ID of the file in the UI, click Documents > Files > File Cabinet. 	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
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 N/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)

Method Description	Method used to load an existing file from the NetSuite file cabinet.
Returns	An existing file.File .
Supported Script Types	Server-side scripts
Governance	10 usage units
Module	N/file Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.id	number string	required	<ul style="list-style-type: none"> ■ Pass one of the following: <ul style="list-style-type: none"> □ Internal ID of the file as a number or a string. □ The relative file path to the file in the file cabinet. For example, 'Images/myImageFile.jpg'. ■ To find the internal ID of the file in the UI, select Documents > Files > File Cabinet. 	Version 2015 Release 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.

Error Code	Message	Thrown If
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

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

```
//Add additional code
...
var fileObj = file.load({
    id: 'Images/myImageFile.jpg'
});
fileObj.description = 'my test file';
var fileId = fileObj.save();
...
//Add additional code
```

file.Encoding

Enum Description	Enumeration that holds the string values for supported character encoding. This enum is used to set the value of the File.encoding property.
	<p> Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Module	N/file Module
Since	Version 2015 Release 2

Values

Value	Character Set
UTF8	Unicode
WINDOWS_1252	Western
ISO_8859_1	Western
GB18030	Chinese Simplified
SHIFT_JIS	Japanese
MAC_ROMAN	Western
GB2312	Chinese Simplified

Value	Character Set
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 [N/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

Enum Description	<p>Enumeration that holds the string values for supported file types. This enum is used to set the value of the File.fileType property.</p> <p>Note that the File.fileType property is read only. Its value must be set with file.create(options).</p> <p>See N/file Module Script Sample for an example.</p> <div style="background-color: #e0f2ff; padding: 10px;"> i Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value. </div>
Module	N/file Module
Since	Version 2015 Release 2

Values

- AUTOCAD
- JSON
- RTF
- BMPIMAGE
- MESSAGERFC
- SMS
- CSV
- MP3
- STYLESHEET
- EXCEL
- MPEGMOVIE
- TAR
- FLASH
- MSPPROJECT
- TIFFIMAGE
- FREEMARKER
- PDF
- VISIO
- GIFIMAGE
- PJPGIMAGE
- WEBAPPPAGE

■ GZIP	■ PLAINTEXT	■ WEBAPPSCRI
■ HTMLDOC	■ PNGIMAGE	■ PT
■ ICON	■ POSTSCRIPT	■ WORD
■ JAVASCRIPT	■ POWERPOINT	■ XMLDOC
■ JPGIMAGE	■ QUICKTIME	■ XSD
		■ ZIP

Syntax

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

```
//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
```

N/format Module

You can use the format module to parse formatted data into strings and to convert strings into a specified format.

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

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

N/format Module Members

Member Type	Name	Return Type / Value Type	Description
Method	format.format(options)	string Date	Takes a raw value and returns a formatted value. <div style="border: 1px solid #0070C0; padding: 5px; background-color: #F0F8FF;"> i Note: This method is overloaded when you format a datetime or datetimetz value. </div>
	format.parse(options)	Date string number	Takes a formatted value and returns a raw value.

Member Type	Name	Return Type / Value Type	Description
			Note: This method is overloaded when you format a datetime or datetimetz value.
Enum	format.Type	enum	Holds the string values for the supported field types. This enum is used to set the value of the options.type parameter.
	format.Timezone	enum	Holds the string values for supported time zone formats. This enum is used to set the value of the options.timezone parameter.

N/format Module Script Sample

The following example parses a string (formatted according to the user preference) to a raw Date Object, and then parses it back to the formatted string.

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

```
/** 
 * @NApiVersion 2.x
 */
require(['N/format'],
function(format) {
    function parseAndFormatDateString() {
        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();
});
```

format.format(options)

Method Description	Method used to format a value from the raw value to its appropriate preference format
--------------------	---

	<p>Note: This method is overloaded when you format a <code>datetime</code> or <code>datetimetz</code> value.</p>
Returns	<p>The formatted value as a string. If a datetime or datetimetz value was given, the Date Object is returned in the user's local app time zone.</p> <p>Note: If an invalid value is given, the original value passed to <code>options.value</code> is returned.</p> <p>Note: For client side scripts, the string returned is based on the user's system time. For server-side scripts, the string returned is based on the system time of the server your NetSuite system is running on.</p>
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/format Module
Since	Version 2015 Release 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.	Version 2015 Release 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	Version 2015 Release 2
<code>options.type</code>	string	required	The field type (either <code>DATETIME</code> or <code>DATETIMETX</code>). Set using the <code>format.Type</code> enum.	Version 2015 Release 2
<code>options.timezone</code>	enum number	optional	The time zone specified for the returned	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<p>string. Set using the format.Timezone enum or key.</p> <p>If a time zone is not specified, the time zone is set based on user preference.</p> <p>If the time zone is invalid, the time zone is set to GMT.</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/format Module Script Sample](#).

```
//Add additional code
...
var formattedDateString = format.format({
    value: parsedDateStringAsRawDateObject,
    type: format.Type.DATE
});
...
//Add additional code
```

format.parse(options)

Method Description	<p>Method used to parse 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.</p> <p>For a <code>datetime</code> or <code>datetimetz</code> value, use this method to convert a Date Object into a string based on the specified timezone.</p> <div style="background-color: #e0f2ff; padding: 10px;"> <p>Note: This method is overloaded when you format a <code>datetime</code> or <code>datetimetz</code> value.</p> </div>
Returns	<p>The parsed value as a Date string number</p> <p>Datetime or datetimetz values are returned as a string.</p> <div style="background-color: #e0f2ff; padding: 10px;"> <p>Note: If the value given is not valid or parsable, the original value passed to <code>options.value</code> is returned.</p> </div>
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/format Module
Since	Version 2015 Release 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>	string	required	The input data to parse.	Version 2015 Release 2
<code>options.type</code>	string	required	The field type (for example, <code>DATE</code> , <code>CURRENCY</code> , <code>INTEGER</code>). Set using the <code>format.Type</code> enum.	Version 2015 Release 2

The table below applies to `datetime` and `datetimetz` values only.

Parameter	Type	Required / Optional	Description	Since
<code>options.value</code>	string	required	The string that contains the date and time information in the specified timezone.	Version 2015 Release 2
<code>options.type</code>	string	required	The field type (either <code>DATETIME</code> or <code>DATETIMETX</code>). Set using the <code>format.Type</code> enum.	Version 2015 Release 2
<code>options.timezone</code>	enum	optional	The time zone represented by the <code>options.value</code> string. Set using the <code>format.Timezone</code> 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.	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var initialFormattedDateString = "07/28/2015";
var parsedDateStringAsRawDateObject = format.parse({
    value: initialFormattedDateString,
    type: format.Type.DATE
});
...
```

```
//Add additional code
```

format.Type

Enum Description	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> .
Module	N/format Module
Since	Version 2015 Release 2

Values

■ CCEXDATE	■ FLOAT	■ POSCURRENCY
■ CCNUMBER	■ FULLPHONE	■ POSFLOAT
■ CCVALIDFRO M	■ FUNCTION	■ POSINTEGER
■ CHECKBOX	■ IDENTIFIER	■ RATE
■ COLOR	■ INTEGER	■ RATEHIGHPRECIS ION
■ CURRENCY	■ MMYYDATE	■ TIME
■ CURRENCY2	■ NONNEGCURRE NCY	■ TIMEOFDAY
■ DATE	■ NONNEGFLOAT	■ TIMETRACK
■ DATETIME	■ PERCENT	■ URL
■ DATETIMETZ	■ PHONE	

Syntax

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 Sample](#).

format.Timezone

Enum Description	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> .
Module	N/format Module

Since	Version 2015 Release 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 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 drop-down.

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

Key	Olson Value	Description
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.
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

Key	Olson Value	Description
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
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.

Key	Olson Value	Description
86	PACIFIC_AUCKLAND: 'Pacific/Auckland'	(GMT+12:00) Auckland, Wellington
87	PACIFIC_TONGATAPU: 'Pacific/Tongatapu'	(GMT+13:00) Nuku'alofa

N/http Module

Use the http module to make HTTP calls from server-side or client-side scripts. On the client-side, this module also provides the ability to make cross-domain HTTP requests using NetSuite servers as a proxies.

All HTTP content types are supported.

Note: The http module does not accept the HTTPS protocol. Use the [N/https Module](#) for that purpose.

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

- [N/http Module Members](#)
- [ClientResponse Object Members](#)
- [ServerRequest Object Members](#)
- [ServerResponse Object Members](#)
- [N/http Module Script Sample](#)

HTTP Header Blacklist

Be aware that certain headers cannot be set manually. The following header types are blacklisted:

<ul style="list-style-type: none"> ▪ Access-Control-Allow-Origin ▪ Allow ▪ Connection ▪ Content-Length ▪ Content-Location ▪ Content-MD5 	<ul style="list-style-type: none"> ▪ Content-Range ▪ Date ▪ Location ▪ Proxy-Authenticate ▪ Retry-After 	<ul style="list-style-type: none"> ▪ Server ▪ Trailer ▪ Via ▪ Warning ▪ WWW-Authenticate
---	--	---

N/http Module Members

Member Type	Name	Return Type / Value Type	Description
Object	http.ClientResponse	read-only Object	Encapsulates the response to an HTTP client request.
	http.ServerRequest	read-only Object	Encapsulates the HTTP request information sent to an HTTP server. For example, a request received by a Suitelet or RESTlet.
	http.ServerResponse	Object	Encapsulates the response from an HTTP server to an

Member Type	Name	Return Type / Value Type	Description
			HTTP request. For example, a response from a Suitelet or RESTlet.
Method	http.delete(options)	http.ClientResponse	Sends an HTTP DELETE request and returns the response.
	http.delete.promise(options)	http.ClientResponse	Sends an HTTP DELETE request asynchronously and returns the response.
	http.get(options)	http.ClientResponse	Sends an HTTP GET request and returns the response.
	http.get.promise(options)	http.ClientResponse	Sends an HTTP GET request asynchronously and returns the response.
	http.post(options)	http.ClientResponse	Sends an HTTP POST request and returns the response.
	http.post.promise(options)	http.ClientResponse	Sends an HTTP POST request asynchronously and returns the response.
	http.put(options)	http.ClientResponse	Sends an HTTP PUT request and returns the response.
	http.put.promise(options)	http.ClientResponse	Sends an HTTP PUT request asynchronously and returns the response.
	http.request(options)	http.ClientResponse	Sends an HTTP request and returns the response.
	http.request.promise(options)	http.ClientResponse	Sends an HTTP request asynchronously and returns the response.
Enum	http.CacheDuration	enum	Holds the string values for supported cache durations. This enum is used to set the value of the <code>ServerResponse.setCdnCacheable(options)</code> property.
	http.Method	enum	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> .
	http.RedirectType	enum	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> .

ClientResponse Object Members

The following members are called on [http.ClientResponse](#).

Member Type	Name	Return Type / Value Type	Description
Property	ClientResponse.body	read-only string	The client response body.
	ClientResponse.code	read-only number	The client response code.
	ClientResponse.headers	read-only Object	The client response headers.

ServerRequest Object Members

The following members are called on the [http.ServerRequest](#) object.

Member Type	Name	Return Type / Value Type	Description
Method	ServerRequest.getLineCount(options)	number	Returns the number of lines in a sublist.
	ServerRequest.getSublistValue(options)	string	Returns the value of a sublist line item.
Property	ServerRequest.body	read-only string	The server request body.
	ServerRequest.files	read-only Object	The server request files.
	ServerRequest.headers	read-only Object	The server request headers.
	ServerRequest.method	http.Method enum	The HTTP method for the server request.
	ServerRequest.parameters	read-only Object	The server request parameters.
	ServerRequest.url	read-only string	The server request URL.

ServerResponse Object Members

The following members are called on the [http.ServerResponse](#) object.

Member Type	Name	Return Type / Value Type	Description
Method	ServerResponse.addHeader(options)	void	Adds a header to the response.
	ServerResponse.getHeader(options)	void	Returns the value of a response header.
	ServerResponse.renderPdf(options)	void	Generates and renders a PDF directly to the response.

Member Type	Name	Return Type / Value Type	Description
	ServerResponse.sendRedirect(options)	void	Sets the redirect URL by resolving to a NetSuite resource.
	ServerResponse.setCdnCacheable(options)	void	Sets CDN caching for a period of time.
	ServerResponse.setHeader(options)	void	Sets the value of a response header.
	ServerResponse.write(options)	void	Writes information (text/xml/html) to the response.
	ServerResponse.writeFile(options)	void	Writes a file to the response.
	ServerResponse.writeLine(options)	void	Writes line information (text/xml/html) to the response.
	ServerResponse.writePage(options)	void	Generates a page.
Property	ServerResponse.headers	Object	The server response headers.

N/http Module Script Sample

Example 1

The following example shows an HTTP GET request for a URL.

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

```
/** 
 * @NApiVersion 2.x
 */
require(['N/http'],
    function(http) {
        function sendGetRequest() {
            var response = http.get({
                url: 'http://www.google.com'
            });
        }
        sendGetRequest();
});
```

Example 2

The following example is designed to redirect to new sales order record, and will set entity to 6. (Assuming there is an entity with number 6, if there's not, then entity will remain blank.)



Note: This sample script uses the define function. Note that you cannot use [Ad Hoc Debugging](#) to step though a define function. You must use [Deployed Debugging](#) to step through this script.

```
/**
 * @NApiVersion 2.x
 * @NScriptType Suitelet
 */
define([ 'N/record', 'N/http' ],
    function(record, http) {
        function onRequest(context) {
            context.response.sendRedirect({
                type: http.RedirectType.RECORD,
                identifier: record.Type.SALES_ORDER,
                parameters: {
                    entity : 6
                }
            });
        }

        return {
            onRequest : onRequest
        };
    });
});
```

http.ClientResponse

Object Description	Encapsulates the response to an HTTP client request. This object is read-only. For a complete list of this object's properties, see ClientResponse Object Members .
Supported Script Types	Server-side scripts
Module	N/http Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
var clientResponse = http.get({
    url: 'http://www.google.com'
});
...
//Add additional code
```

ClientResponse.body

Property Description	The client response body. This property is read-only.
Type	string
Module	N/http Module
Since	Version 2015 Release 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 Sample](#).

```
//Add additional code
...
var response = http.get({
    url: 'http://www.google.com'
});
log.debug({
    title: 'Client Response Body',
    details: http.response.body
});
...
//Add additional code
```

ClientResponse.code

Property Description	The client response code. This property is read-only.
Type	number
Module	N/http Module
Since	Version 2015 Release 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 Sample](#).

```
//Add additional code
...
var response = http.get({
    url: 'http://www.google.com'
});
log.debug({
    title: 'Client Response Code',
    details: http.response.code
});
...
//Add additional code
```

ClientResponse.headers

Property Description	The response header or headers. This property is read-only.
Type	object
Module	N/http Module
Since	Version 2015 Release 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 Sample](#).

```
//Add additional code
...
var response = http.get({
    url: 'http://www.google.com'
});
log.debug({
    title: 'Client Response Header',
    details: http.response.headers
});
...
//Add additional code
```

http.ServerRequest

Object Description	Encapsulates the HTTP request information to an HTTP server. For example, a request received by a Suitelet or RESTlet. This object is read-only. For a complete list of this object's methods and properties, see ServerRequest Object Members .
Supported Script Types	Server-side scripts
Module	N/http Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
serverRequest.getLineCount({
    group: 'sublistId'
});
...
//Add additional code
```

ServerRequest.getLineCount(options)

Method Description	Method used to return the number of lines in a sublist.
Returns	The number of lines in a sublist as a number.
Supported Script Types	Server-side scripts
Governance	None
Module	N/http Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.group	string	required	The sublist internal ID.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax



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

```
//Add additional code
...
serverRequest.getLineCount({
    group: 'sublistId'
});
...
//Add additional code
```

ServerRequest.getSublistValue(options)

Method Description	Method used to return the value of a sublist line item.
Returns	The value of the sublist line item as a string.
Supported Script Types	Server-side scripts
Governance	None
Module	N/http Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.group	string	required	The sublist internal ID.	Version 2015 Release 2
options.name	string	required	The sublist line item ID (name of the field).	Version 2015 Release 2
options.line	string	required	The sublist line number. Note: Sublist index starts at 0.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax



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

```
//Add additional code
...
serverRequest.getSublistValue({
    group: 'item',
    name: 'amount',
    line: '2'
});
...
//Add additional code
```

ServerRequest.body

Property Description	The server request body. This property is read-only.
Type	string
Module	N/http Module
Since	Version 2015 Release 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 Sample](#).

```
//Add additional code
...
log.debug({
    title: 'Server Request Body',
    details: http.request.body
});
...
//Add additional code
```

ServerRequest.files

Property Description	The server request files.
----------------------	---------------------------

	This property is read-only.
Type	Object
Module	N/http Module
Since	Version 2015 Release 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 Sample](#).

```
//Add additional code
...
log.debug({
    title: 'Server Request Files',
    details: http.request.files
});
...
//Add additional code
```

ServerRequest.headers

Property Description	This object represents a series of key/value pairs. Each pair represents a server request header name and its value. Typically, this object encapsulates two iterations of each header name: one in lower case and another in title case. This behavior is designed so that you can use either lower case or title case when you reference a header. However, the existence of title-case iterations of header names is not guaranteed. For best results, refer to header names using all lower-case letters (and hyphens, when applicable). This property is read-only.
Type	Object
Module	N/http Module
Since	Version 2015 Release 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 Sample](#).

```
//Add additional code
...
log.debug({
    title: 'Server Request Headers',
    details: http.request.headers
});
...
//Add additional code
```

ServerRequest.method

Property Description	The server request http method. This property is read-only.
Type	<code>http.Method</code> enum
Module	N/http Module
Since	Version 2015 Release 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 Sample](#).

```
//Add additional code
...
log.debug({
    title: 'Server Request Method',
    details: http.request.method
});
...
//Add additional code
```

ServerRequest.parameters

Property Description	The server request parameters.
----------------------	--------------------------------

	This property is read-only.
Type	Object
Module	N/http Module
Since	Version 2015 Release 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 Sample](#).

```
//Add additional code
...
log.debug({
    title: 'Server Request Parameters',
    details: http.request.parameters
});
...
//Add additional code
```

ServerRequest.url

Property Description	The server request URL. This property is read-only.
Type	string
Module	N/http Module
Since	Version 2015 Release 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 Sample](#).

```
//Add additional code
```

```

...
log.debug({
    title: 'Server Request URL',
    details: http.request.url
});
...
//Add additional code

```

http.ServerResponse

Object Description	Encapsulates the response to an incoming http request from an HTTP server. For example, a response from a Suitelet or RESTlet. For a complete list of this object's methods and properties, see ServerResponse Object Members .
Supported Script Types	Server-side scripts
Module	N/http Module
Since	Version 2015 Release 2

Syntax



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

```

//Add additional code
...
serverResponse.addHeader({
    name: 'Accept-Language',
    value: 'en-us',
});
...
//Add additional code

```

ServerResponse.addHeader(options)

Method Description	Method used to add a header to the response. If the same header has already been set, this method adds another line for that header. For example: <pre>Vary: 'Accept-Language' Vary: 'Accept-Encoding'</pre>
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/http Module

Since	Version 2015 Release 2
-------	------------------------

Parameters

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	The name of the header.	Version 2015 Release 2
options.value	string	required	The value used to set the header.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.
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 Sample](#).

```
//Add additional code
...
serverResponse.addHeader({
    name: 'Accept-Language',
    value: 'en-us',
});
...
//Add additional code
```

ServerResponse.getHeader(options)

Method Description	Method used to return the value or values of a response header. If multiple values are assigned to the header name, the values are returned as an Array.
Returns	string string[]
Supported Script Types	Server-side scripts
Governance	None
Module	N/http Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	The name of the header.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax



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

```
//Add additional code
...
serverResponse.getHeader({
    name: 'Accept-Language'
});
...
//Add additional code
```

ServerResponse.sendRedirect(options)

Method Description	Method used to set the redirect URL by resolving to a NetSuite resource. For example, you could use this method and your own parameters to make a redirect to a url for associated records, such as a redirect to a new sales order page for a particular entity.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/http Module
Since	Version 2015 Release 2

Parameters



Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The type of resource redirected to. Set	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			this value using the http.RedirectType enum.	
options.identifier	number string	required	<p>The primary ID for the resource.</p> <ul style="list-style-type: none"> ■ If redirecting to a media item (for example, an image or PDF file), pass in the file id. ■ If redirecting to a record, pass in the record type using the record.Type enum. ■ If redirecting to a RESTlet, pass in the script ID as a number or string. ■ If redirecting to a tasklink, pass in the task ID. For a list of supported task IDs, see the help topic Task IDs. ■ If redirecting to a Suitelet, pass in the script ID. 	Version 2015 Release 2
options.id	number string	optional	The secondary ID for this resource. If the resource type is a Suitelet or RESTlet, pass in the deployment ID.	Version 2015 Release 2
options.editMode	boolean true false	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>The default value is <code>false</code> – returns the record in view mode and not edit mode.</p>	Version 2015 Release 2
options.parameters	object	optional	Additional URL parameters as name/value pairs.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.
SSS_INVALID_URL_CATEGORY	The URL category must be one of RECORD, TASKLINK or SUITELET.	The input for options.type is not valid.
SSS_INVALID_TASK_ID	The task ID: {id} is not valid. Please refer to the documentation for a list of supported task IDs.	The type is set to tasklink, and an invalid task ID is input for options.identifier.
SSS_INVALID_RECORD_TYPE	Type argument {type} is not a valid record or is not available in your account. Please see the documentation for a list of supported record types.	The redirect type is set to record, and an invalid record type is input for options.identifier.
SSS_INVALID_SCRIPT_ID_1	You have provided an invalid script id or internal id: {id}	The type is set to Suitelet or RESTlet, and an invalid script ID or invalid deployment ID is input for options.identifier or options.id.

Syntax



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

```
//Add additional code
...
myServerResponseObj.sendRedirect({
    type: http.RedirectType.RECORD,
    identifier: record.Type.SALES_ORDER,
    parameters: {entity: 8}
});
...
//Add additional code
```

ServerResponse.setHeader(options)

Method Description	Method used to set the value of a response header.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/http Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	The name of the header.	Version 2015 Release 2
options.value	string	required	The value used to set the header.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.
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 Sample](#).

```
//Add additional code
...
serverResponse.setHeader({
    name: 'Accept-Language',
    value: 'en-us',
});
...
//Add additional code
```

ServerResponse.renderPdf(options)

Method Description	Method used to generate and render a PDF directly to the response.
Returns	Void
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/http Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.xmlString	string	required	Content of the pdf.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax



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

```
//Add additional code
...
serverResponse.renderPDF({
    xmlString:'<?xml version="1.0"?>\n<!DOCTYPE pdf PUBLIC "-//big.faceless.org//report" "report-1.1.dtd">\n<pdf>\n<body font-size="18">\nHello World!\n</body>\n</pdf>'
});
...
//Add additional code
```

ServerResponse.setCdnCacheable(options)

Method Description	Method used to set CDN caching for a period of time.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/http Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.type	enum	required	The value of the caching duration. Set using the http.CacheDuration enum.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax

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

```
//Add additional code
...
serverResponse.setCdnCacheable({
    type: http.CacheDuration.MAX
});
...
//Add additional code
```

ServerResponse.write(options)

Method Description	Method used to write information to the response.  Note: This method accepts only strings. To pass in a file, you can use ServerResponse.writeFile(options) .
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/http Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.output	string	required	The output string being written.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: argument: {param name}	A required parameter is not passed.

Error Code	Message	Thrown If
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 Sample](#).

```
//Add additional code
...
serverResponse.write({
    output: 'Hello World'
});
...
//Add additional code
```

ServerResponse.writeFile(options)

Method Description	Method used to write a file to the response.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/http Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.file	file.File	required	A file.File Object that encapsulates the file to be written.	Version 2015 Release 2
options.inline	boolean true false	optional	Determines whether the file is inline. If true, the file is inline. The default value is false.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.
WRONG_PARAMETER_TYPE	{param name}	The value input for options.file is not a file.File 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 Sample](#).

```
//Add additional code
...
serverResponse.writeFile({
    file: myFileObj,
    isInline: true
});
...
//Add additional code
```

ServerResponse.writeLine(options)

Method Description	Method used to write line information to the response.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/http Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.output	string	required	The output string being written.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.
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 Sample](#).

```
//Add additional code
```

```

...
serverResponse.writeLine({
    output: 'this is a sample string'
});
...
//Add additional code

```

ServerResponse.writePage(options)

Method Description	Method used to generate a page.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/http Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.pageObject	serverWidget.Assistant serverWidget.Form serverWidget.List	required	A standalone page object in the form of an assistant, form or list.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax



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

```

//Add additional code
...
var myPageObj = serverWidget.createList({
    title: 'Simple List'
});

ServerResponse.writePage({
    pageObject: myPageObj
});
...

```

```
//Add additional code
```

ServerResponse.headers

Property Description	The server response headers. This property is read-only.
Type	Object 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>
Module	N/http Module
Since	Version 2015 Release 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 Sample](#).

```
//Add additional code
...
if serverResponse.headers.ContentType === 'text/plain'
return true

log.debug({
  title: "Server Response Headers",
  details: ServerResponse.headers
});
...
//Add additional code
```

http.get(options)

Method Description	Method used to send an HTTP GET request and return the response
Returns	http.ClientResponse
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/http Module

Since	Version 2015 Release 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	Version 2015 Release 2
options.headers	object	optional	The HTTP headers.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax

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

```
//Add additional code
...
var response = http.get({
    url: 'http://www.google.com'
});
...
//Add additional code
```

http.get.promise(options)

Method Description	Method used to send an HTTP GET request asynchronously and return the response
	 Note: For information about the parameters and errors thrown for this method, see http.get(options) . For additional information on promises, see Promise object .
Returns	A http.ClientResponse object.
Synchronous Version	http.get(options)
Supported Script Types	All client-side scripts

Governance	10 units
Module	N/http Module
Since	Version 2015 Release 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
...
http.get.promise({
    url: 'http://www.google.com'
})
    .then(function(response){
        log.debug({
            title: 'Response',
            details: response
        });
    })
    .catch(function onRejected(reason) {
        log.debug({
            title: 'Invalid Get Request: ',
            details: reason
        });
    })
...
//Add additional code
```

http.delete(options)

Method Description	<p>Method used to send an HTTP DELETE request and return the response.</p> <p> Important: If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs.</p> <p> Note: This method does not include an <code>options.body</code> parameter. Postdata is not required when the HTTP method is a DELETE request.</p>
Returns	<code>http.ClientResponse</code>
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/http Module
Since	Version 2015 Release 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	Version 2015 Release 2
options.headers	object	optional	The HTTP headers.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax

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

```
//Add additional code
...
var response = http.delete({
    url: 'http://www.mytestwebsite.com'
});
...
//Add additional code
```

http.delete.promise(options)

Method Description	Method used to send an HTTP DELETE request asynchronously and return the response.
	 Note: For information about the parameters and errors thrown for this method, see http.delete(options) . For additional information on promises, see Promise object .
Returns	A http.ClientResponse object
Synchronous Version	http.delete(options)
Supported Script Types	All client-side scripts
Governance	10 units
Module	N/http Module
Since	Version 2015 Release 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
...
http.delete.promise({
    url: 'http://www.mytestwebsite.com'
})
    .then(function(response){
        log.debug({
            title: 'Response',
            details: response
        });
    })
    .catch(function onRejected(reason) {
        log.debug({
            title: 'Invalid Request: ',
            details: reason
        });
    })
...
//Add additional code
```

http.request(options)

Method Description	Method used to send an HTTP request and return the response.
	<p>⚠ Important: If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs.</p>
Returns	A http.ClientResponse object
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/http Module
Since	Version 2015 Release 2

Parameters

i 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 http.Method enum.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.url	string	required	The HTTP URL being requested	Version 2015 Release 2
options.body	string object	optional	The POST data if the method is POST. Note: If the method is DELETE, this body data is ignored.	
options.headers	object	optional	An object containing request headers.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax



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

```
//Add additional code
...
var response = http.request({
    method: http.Method.GET,
    url: 'http://www.google.com'
});
...
//Add additional code
```

http.request.promise(options)

Method Description	Method used to send an HTTP request asynchronously and return the response. Note: For information about the parameters and errors thrown for this method, see http.request(options) . For additional information on promises, see Promise object .
Returns	A http.ClientResponse object
Synchronous Version	http.request(options)

Supported Script Types	All client-side scripts
Governance	10 units
Module	N/http Module
Since	Version 2015 Release 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
...
http.request.promise({
    method: http.Method.GET,
    url: 'http://www.google.com'
})
    .then(function(response){
        log.debug({
            title: 'Response',
            details: response
        });
    })
    .catch(function onRejected(reason) {
        log.debug({
            title: 'Invalid Request: ',
            details: reason
        });
    })
...
//Add additional code
```

http.post(options)

Method Description	Method used to send an HTTP POST request and return the response.
Returns	A http.ClientResponse object
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/http Module
Since	Version 2015 Release 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	Version 2015 Release 2
options.body	string object	required	The POST data.	Version 2015 Release 2
options.headers	object	optional	The HTTP headers.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax



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

```
//Add additional code
...
var response = https.post({
    url: 'http://www.google.com',
    body: mypostDataObj
});
...
//Add additional code
```

http.post.promise(options)

Method Description	Method used to send an HTTP POST request asynchronously and return the response.
	 Note: For information about the parameters and errors thrown for this method, see http.post(options) . For additional information on promises, see Promise object .
Returns	A http.ClientResponse object
Synchronous Version	http.post(options)

Supported Script Types	All client-side scripts
Governance	10 units
Module	N/http Module
Since	Version 2015 Release 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
...
http.post.promise({
    url: 'http://www.google.com',
    body: myPostDataObj
})
    .then(function(response){
        log.debug({
            title: 'Response',
            details: response
        });
    })
    .catch(function onRejected(reason) {
        log.debug({
            title: 'Invalid Request: ',
            details: reason
        });
    })
...
//Add additional code
```

http.put(options)

Method Description	Method used to send an HTTP PUT request and return the response.
	⚠ Important: If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs.
Returns	http.ClientResponse object
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/http Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.url	string	required	The HTTP URL being requested	Version 2015 Release 2
options.body	string object	required	The PUT data.	Version 2015 Release 2
options.headers	object	optional	The HTTP headers.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax

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

```
//Add additional code
...
var response = http.put({
    url: 'http://www.google.com',
    body: myDataObj,
    headers: headerObj
});
...
//Add additional code
```

http.put.promise(options)

Method Description	Method used to send an HTTP PUT request asynchronously and return the response.
	i Note: For information about the parameters and errors thrown for this method, see http.put(options) . For additional information on promises, see Promise object .
Returns	http.ClientResponse object
Synchronous Version	http.put(options)
Supported Script Types	All client-side scripts
Governance	10 units

Module	N/http Module
Since	Version 2015 Release 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
...
http.put.promise({
    url: 'http://www.google.com',
    body: myDataObj,
    headers: headerObj
})
    .then(function(response){
        log.debug({
            title: 'Response',
            details: response
        });
    })
    .catch(function onRejected(reason) {
        log.debug({
            title: 'Invalid Request: ',
            details: reason
        });
    })
...
//Add additional code
```

http.CacheDuration

Enum Description	Holds the string values for supported cache durations. This enum is used to set the value of the <code>ServerResponse.setCdnCacheable(options)</code> property.
Module	N/http Module

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](#).

Syntax



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

```
//Add additional code
...
ServerResponse.setCdnCacheable({
    type: http.CacheDuration.MAX
});
...
//Add additional code
```

http.Method

Enum Description	Holds the string values for supported HTTP requests. This enum is used to set the value of http.request(options) and ServerRequest.method . i Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Module	N/http Module

Values

- DELETE
- GET
- 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](#).

```
//Add additional code
...
var response = http.request({
    method: http.Method.GET,
    url: 'http://www.google.com'
});
...
```

```
//Add additional code
```

http.RedirectType

Enum Description	Holds the string values for supported NetSuite resources that you can redirect to. This enum is used to set the value of the <code>type</code> argument for <code>ServerResponse.sendRedirect(options)</code> . Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Module	N/http Module

Values

- MEDIAITEM
- RECORD
- RESTLET
- SUITELET
- TASKLINK

Syntax



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

```
//Add additional code
...
myServerResponseObj.sendRedirect({
    type: http.RedirectType.RECORD,
    identifier: record.Type.SALES_ORDER,
    parameters: {entity: 6}
});...
//Add additional code
```

N/https Module

Load the https module when you need to manage content sent to a third party via HTTPS calls. This module encapsulates all the functionality of the [N/http Module](#), but does not allow the HTTP protocol. You can make HTTPS calls from client and server-side scripts.

SecureString functionality is supported only in server-side scripts. You can also use this functionality to perform various string transformations using methods that hash, encode, or append another string.

You can use this module to encode binary content or access a handle to the value in a NetSuite credential field.

When the https module is used, SuiteScript also loads the [N/crypto Module](#) and [N/encode Module](#).



Important: TLS 1.0, 1.1, and 1.2 are currently supported. NetSuite recommends that you use TLS 1.2 for https requests. For more information, see the help topics [FAQ: Transport Layer Security \(TLS\) Deprecations](#), specifically [SuiteScript and TLS](#).



Important: NetSuite supports the same list of trusted third-party certificate authorities (CAs) as Microsoft. For a list of these CAs, see <http://social.technet.microsoft.com/wiki/contents/articles/31634.microsoft-trusted-root-certificate-program-participants-v-2016-april.aspx>



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

- N/https Module Members
- SecureString Object Members
- ClientResponse Object Members
- ServerResponse Object Members
- ServerRequest Object Members
- N/https Module Script Sample

Header Blacklist

Be aware that certain headers cannot be set manually. The following header types are blacklisted:

<ul style="list-style-type: none"> ▪ Access-Control-Allow-Origin ▪ Allow ▪ Connection ▪ Content-Length ▪ Content-Location ▪ Content-MD5 	<ul style="list-style-type: none"> ▪ Content-Range ▪ Date ▪ Location ▪ Proxy-Authenticate ▪ Retry-After 	<ul style="list-style-type: none"> ▪ Server ▪ Trailer ▪ Via ▪ Warning ▪ WWW-Authenticate
---	--	---

N/https Module Members

Member Type	Name	Return Type / Value Type	Description
Object	https.SecureString	Object	Encapsulates data that may be sent to a third-party via an HTTPS call.
	https.ClientResponse	read-only Object	Encapsulates the response to an HTTPS client request.
	https.ServerRequest	read-only Object	Encapsulates the HTTPS request information sent to an HTTPS server. For example, a request received by a Suitelet or RESTlet.
	https.ServerResponse	Object	Encapsulates the response from an HTTPS server to an HTTPS request. For example, a response from a Suitelet or RESTlet.

Member Type	Name	Return Type / Value Type	Description
Method	https.createSecureKey(options)	Object	Creates a key for the contents of a credential field.
	https.createSecureKey.promise(options)	Object	Creates a key asynchronously for the contents of a credential field.
	https.createSecureString(options)	Object	Creates an https.SecureString object.
	https.createSecureString.promise(options)	Object	Creates an https.SecureString object asynchronously.
	https.delete(options)	https.ClientResponse	Sends an HTTPS DELETE request and returns the response.
	https.delete.promise(options)	https.ClientResponse	Sends an HTTPS DELETE request asynchronously and returns the response.
	https.get(options)	https.ClientResponse	Sends an HTTPS GET request and returns the response.
	https.get.promise(options)	https.ClientResponse	Sends an HTTPS GET request asynchronously and returns the response.
	https.post(options)	https.ClientResponse	Sends an HTTPS POST request and returns the response.
	https.post.promise(options)	https.ClientResponse	Sends an HTTPS POST request asynchronously and returns the response.
	https.put(options)	https.ClientResponse	Sends an HTTPS PUT request and returns the response.
	https.put.promise(options)	https.ClientResponse	Sends an HTTPS PUT asynchronously request and returns the response.
	https.request(options)	https.ClientResponse	Sends an HTTPS request and returns the response.
	https.request.promise(options)	https.ClientResponse	Sends an HTTPS request asynchronously and returns the response.
Enum	https.CacheDuration	enum	Holds the string values for supported cache durations. This enum is used to set the value of the ServerResponse.setCdnCacheable(options) property.
	https.Encoding	enum	Holds the string values for supported encoding types. This enum is used to set the value of parameters in

Member Type	Name	Return Type / Value Type	Description
			SecureString.appendString(options), SecureString.convertEncoding(options), https.createSecureString(options).
	https.Method	enum	Holds the string values for supported HTTP requests. This enum is used to set the value of https.request(options) and ServerRequest.method.

SecureString Object Members

The following members are called on https.SecureString.

Member Type	Name	Return Type / Value Type	Description
Method	SecureString.appendSecureString(options)	https.SecureString	Appends a passed in https.SecureString to another https.SecureString.
	SecureString.appendString(options)	https.SecureString	Appends a passed in string to a https.SecureString.
	SecureString.convertEncoding(options)	https.SecureString	Changes the encoding of a https.SecureString.
	SecureString.hash(options)	https.SecureString	Produces the https.SecureString as a hash.
	SecureString.hmac(options)	https.SecureString	Produces the https.SecureString as an hmac.

ClientResponse Object Members

The following members are called on http.ClientResponse.

Member Type	Name	Return Type / Value Type	Description
Property	ClientResponse.body	read-only string	The response body.
	ClientResponse.code	read-only number	The response code.
	ClientResponse.headers	read-only Object	The response body.

ServerRequest Object Members

The following members are called on the http.ServerRequest.

Member Type	Name	Return Type / Value Type	Description
Method	ServerRequest.getLineCount(options)	number	Returns the number of lines in a sublist.
	ServerRequest.getSublistValue(options)	string	Returns the value of a sublist line item.
Property	ServerRequest.body	read-only string	The server request body
	ServerRequest.files	read-only Object	The server request files.
	ServerRequest.headers	read-only Object	The server request headers.
	ServerRequest.method	https.Method enum	The HTTPS method for the server request.
	ServerRequest.parameters	read-only Object	The server request parameters.
	ServerRequest.url	read-only string	The server request URL.

ServerResponse Object Members

The following members are called on the [http.ServerResponse](#).

Member Type	Name	Return Type / Value Type	Description
Method	ServerResponse.addHeader(options)	void	Adds a header to the response
	ServerResponse.getHeader(options)	void	Returns the value of a response header
	ServerResponse.renderPdf(options)	void	Generates and renders a PDF directly to the response
	ServerResponse.sendRedirect(options)	void	Sets the redirect URL by resolving to a NetSuite resource
	ServerResponse.setCdnCacheable(options)	void	Sets CDN caching for a period of time.
	ServerResponse.setHeader(options)	void	Sets the value of a response header.
	ServerResponse.write(options)	void	Writes information (text/xml/html) to the response.
	ServerResponse.writeFile(options)	void	Writes a file to the response.
	ServerResponse.writeLine(options)	void	Writes line information (text/xml/html) to the response.
	ServerResponse.writePage(options)	void	Generates a page.

Member Type	Name	Return Type / Value Type	Description
Property	ServerResponse.headers	Object	The server response headers.

N/https Module Script Sample

The following example uses a GUID to generate a secure token and a secret key. Note that you must replace the GUID with one specific to your account.

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

```
/** 
 * @NApiVersion 2.x
 */
require(['N/https'],
    function(https) {
        function createSecureString() {
            var passwordGuid = '284CFB2D225B1D76FB94D150207E49DF';
            var secureToken = https.createSecureString({
                input: passwordGuid
            });
            var secretKey = https.createSecretKey({
                encoding: https.Encoding.HEX,
                guid: passwordGuid
            });
            secureToken = secureToken.hmac({
                algorithm: crypto.HashAlg.SHA256,
                key: secretKey
            });
        }
        createSecureString();
    });

```

The following example is a Suitelet sample that shows creating a form field that generates a GUID..

For more information about credential fields, see [Form.addCredentialField\(options\)](#).

Note: The values for `restrictToDomains`, `restrictToScriptIds`, and `baseUrl` in this sample are placeholders. You must replace them with valid values from your NetSuite account.

Note: This sample uses the `define` function. The NetSuite Debugger cannot step though a `define` function. If you need to step through your code in the NetSuite Debugger, you must use a `require` function.

```
/** 
 * @NApiVersion 2.x
 * @NScriptType Suitelet
 */
define(['N/ui/serverWidget', 'N/https'],
```

```

function(ui, https) {
    function onRequest(option) {
        if (option.request.method === 'GET') {
            var form = ui.createForm({
                title: 'Password Form'
            });
            form.addCredentialField({
                id: 'password',
                label: 'Password',
                restrictToDomains: ['system.netsuite.com'],
                restrictToCurrentUser: false,
                restrictToScriptIds: 'customscript_my_script'
            });
            form.addSubmitButton();
            option.response.writePage({
                pageObject: form
            });
        } else {
            // Request to an existing suitelet with credentials
            var passwordGuid = option.request.parameters.password;
            var baseUrl = 'https://system.netsuite.com/app/site/hosting/scriptlet.nl?script='
            + 995 + '&deploy=1&compid=1326288&h=397b6b0d3a4170b26735';
            var url = baseUrl + '&pwd={' + passwordGuid + '}';
            var secureStringUrl = https.createSecureString({
                input: url
            });
            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

Object Description	Encapsulates a request string, such as a fragment of sensitive data that is going to be sent to a third party. This object is needed when you create a securestring, put your data in it, and encode it a particular way. For a complete list of this object's methods, see SecureString Object Members .
---------------------------	---

Supported Script Types	Server-side scripts
Module	N/https Module
Since	Version 2015 Release 2

Syntax

⚠ 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)

Method Description	Method used to append a passed in <code>https.SecureString</code> to another <code>https.SecureString</code> .
Returns	<code>https.SecureString</code>
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 2

Parameters

i Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
<code>options.secureString</code>	<code>https.SecureString</code> object	required	The <code>https.SecureString</code> 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)

Method Description	Method used to append a passed string to an https.SecureString .
Returns	https.SecureString
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 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	https.Encoding enum	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: '48656c6f20776f726c640d0a',
    encoding: https.Encoding.HEX});
...
//Add additional code

```

SecureString.convertEncoding(options)

Method Description	Changes the encoding of a https.SecureString
Returns	https.SecureString
Supported Script Types	Server-side scripts

Governance	None
Module	N/https Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
options.toEncoding	https.Encoding	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)

Method Description	Hashes an https.SecureString object
Returns	https.SecureString
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
options.algorithm	crypto.Hash enum	required	The hash algorithm. Set the value using the crypto.Hash 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)

Method Description	Produces the <code>securestring</code> as an <code>hmac</code> .
Returns	https.SecureString
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 2

Parameters



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

Parameter	Type	Required / Optional	Description
<code>options.algorithm</code>	<code>crypto.Hash</code> enum	required	The hash algorithm. Set by the <code>crypto.Hash</code> enum.
<code>options.key</code>	<code>crypto.SecretKey</code>	required	A key returned from <code>https.createSecureKey(options)</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/https Module Script Sample](#).

```
//Add additional code
...
secureToken = secureToken.hmac({
    algorithm: crypto.HashAlg.SHA256,
    key: secretKey
});
...
//Add additional code
```

https.createSecureKey(options)

Method Description	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
Returns	<code>crypto.SecretKey</code>
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 2

Parameters

i Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
<code>options.encoding</code>	<code>https.Encoding</code> enum	optional	Specifies the encoding for the SecureKey.	Version 2015 Release 2
<code>options.guid</code>	string	required	A GUID used to generate a secret key. The GUID can resolve to either data or metadata.	Version 2015 Release 2

Syntax

⚠ 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)

Method Description	Creates and returns a <code>crypto.SecretKey</code> object asynchronously.
---------------------------	--

	Note: For information about the parameters and errors thrown for this method, see https.createSecureKey(options) . For additional information on promises, see Promise object .
Returns	A crypto.SecretKey object
Synchronous Version	https.createSecureKey(options)
Supported Script Types	All client-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 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 secretKey = https.createSecureKey.promise({
    encoding: https.Encoding.HEX,
    guid: '284CFB2D225B1D76FB94D150207E49DF'
});

...
//Add additional code
```

https.createSecureString(options)

Method Description	Creates and returns an https.SecureString .
Returns	https.SecureString
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 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 enum	optional	Identifies the encoding that the input string uses. The default value is UTF_8	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)

Method Description	Creates and returns an https.SecureString asynchronously.
	i Note: For information about the parameters and errors thrown for this method, see https.createSecureString(options) . For additional information on promises, see Promise object .
Returns	SecureTokenResolver
Synchronous Version	https.createSecureString(options)
Supported Script Types	All client-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 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 secureToken = https.createSecureString.promise({
    input: passwordGuid
});
...
```

```
//Add additional code
```

https.ClientResponse

Object Description	Encapsulates the response to an HTTPS client request. This object is read-only. For a complete list of this object's properties, see ClientResponse Object Members .
Supported Script Types	Server-side scripts
Module	N/https Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var clientResponse = https.get({
    url: 'https://www.testwebsite.com'
});
...
//Add additional code
```

ClientResponse.body

Property Description	The client response body. This property is read-only.
Type	string
Module	N/https Module
Since	Version 2015 Release 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 Sample](#).

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

```

var response = https.get({
    url: 'https://www.testwebsite.com'
});
log.debug({
    title: 'Client Response Body',
    details: https.response.body
});
...
//Add additional code

```

ClientResponse.code

Property Description	The client response code. This property is read-only.
Type	number
Module	N/https Module
Since	Version 2015 Release 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 Sample](#).

```

//Add additional code
...
var response = https.get({
    url: 'https://www.testwebsite.com'
});
log.debug({
    title: 'Client Response Code',
    details: https.response.code
});
...
//Add additional code

```

ClientResponse.headers

Property Description	The response header or headers. This property is read-only.
Type	object
Module	N/https Module

Since	Version 2015 Release 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 Sample](#).

```
//Add additional code
...
var response = https.get({
    url: 'https://www.testwebsite.com'
});
log.debug({
    title: 'Client Response Header',
    details: https.response.headers
});
...
//Add additional code
```

https.ServerRequest

Object Description	Encapsulates the incoming https request information for an HTTPS server. This object is read-only. For a complete list of this object's methods and properties, see ServerRequest Object Members .
Supported Script Types	Server-side scripts
Module	N/https Module
Since	Version 2015 Release 2

Syntax

⚠ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [https.ServerRequest](#).

```
//Add additional code
...
serverRequest.getLineCount({
    group: 'sublistId'
});
...
//Add additional code
```

ServerRequest.getLineCount(options)

Method Description	Method used to return the number of lines in a sublist.
Returns	The number of lines in a sublist as a number.
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.group	string	required	The sublist internal ID.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax



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

```
//Add additional code
...
serverRequest.getLineCount({
    group: 'sublistId'
});
...
//Add additional code
```

ServerRequest.getSublistValue(options)

Method Description	Method used to return the value of a sublist line item.
Returns	The value of the sublist line item as a string.
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module

Since	Version 2015 Release 2
-------	------------------------

Parameters

Parameter	Type	Required / Optional	Description	Since
options.group	string	required	The sublist internal ID.	Version 2015 Release 2
options.name	string	required	The name of the field.	Version 2015 Release 2
options.line	string	required	The sublist line number.  Note: Sublist index starts at 0.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax



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

```
//Add additional code
...
serverRequest.getSublistValue({
    group: 'item',
    name: 'amount',
    line: '2'
});
...
//Add additional code
```

ServerRequest.body

Property Description	The server request body. This property is read-only.
Type	string
Module	N/https Module
Since	Version 2015 Release 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: https.request.body
});
...
//Add additional code
```

ServerRequest.files

Property Description	The server request files. This property is read-only.
Type	Object
Module	N/https Module
Since	Version 2015 Release 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 Files',
    details: https.request.files
});
...
//Add additional code
```

ServerRequest.headers

Property Description	This object represents a series of key/value pairs. Each pair represents a server request header name and its value. Typically, this object encapsulates two iterations of each header name: one in lower case and another in title case. This behavior is designed so that you can use either lower case or title case when you reference a header. However, the existence of title-case iterations of header names is not guaranteed. For best results, refer to header names using all lower-case letters (and hyphens, when applicable). This property is read-only.
Type	Object
Module	N/https Module
Since	Version 2015 Release 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: https.request.headers
});
...
//Add additional code
```

ServerRequest.method

Property Description	The server request https method. This property is read-only.
Type	enum
Module	N/https Module
Since	Version 2015 Release 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 [Parameters](#).

```
//Add additional code
...
log.debug({
    title: 'Server Request Method',
    details: https.request.method
});
...
//Add additional code
```

ServerRequest.parameters

Property Description	The server request parameters. This property is read-only.
Type	Object
Module	N/https Module
Since	Version 2015 Release 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: https.request.parameters
});
...
//Add additional code
```

ServerRequest.url

Property Description	The server request URL. This property is read-only.
Type	string

Module	N/https Module
Since	Version 2015 Release 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: https.request.url
});
...
//Add additional code
```

https.ServerResponse

Object Description	Encapsulates the response to an incoming http request from an HTTP server. For example, a response from a Suitelet or RESTlet. For a complete list of this object's methods and properties, see ServerResponse Object Members .
Supported Script Types	Server-side scripts
Module	N/https Module
Since	Version 2015 Release 2

Syntax



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)

Method Description	Method used to add a header to the response. If the same header has already been set, this method adds another line for that header. For example:
	<pre>Vary: 'Accept-Language' Vary: 'Accept-Encoding'</pre>
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	The name of the header.	Version 2015 Release 2
options.value	string	required	The value used to set the header.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.
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)

Method Description	Method used to return the value or values of a response header. If multiple values are assigned to the header name, the values are returned as an Array.
Returns	string string[]
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	The name of the header.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax

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

```
//Add additional code
...
serverResponse.getHeader({
    name: 'Accept-Language'
});
...
//Add additional code
```

ServerResponse.sendRedirect(options)

Method Description	Method used to set the redirect URL by resolving to a NetSuite resource. For example, you could use this method and your own parameters to make a redirect to a url for associated records, such as a redirect to a new sales order page for a particular entity.
Returns	Void
Supported Script Types	Server-side scripts

Governance	None
Module	N/https Module
Since	Version 2015 Release 2

Parameters

 **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 base type for this resource. Use one of the following values: <ul style="list-style-type: none">■ RECORD■ TASKLINK■ SUITELET	Version 2015 Release 2
options.identifier	string	required	The primary ID for this resource. <ul style="list-style-type: none">■ If the base type is RECORD, input the record type as listed on the Records Browser.■ If the base type is TASKLINK, input the task ID.■ If the base type is SUITELET, input the script ID.	Version 2015 Release 2
options.id	string	optional	The secondary ID for this resource. If the base type is SUITELET, input the deployment ID.	Version 2015 Release 2
options.editMode	boolean true false	optional	Applicable when redirecting to a record resource. Specifies whether to return a URL for a record in edit mode or view mode. The default value is false – returns the record in view mode and not edit mode.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.parameters	object	optional	Additional URL parameters as name/value pairs.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.
SSS_INVALID_URL_CATEGORY	The URL category must be one of RECORD, TASKLINK or SUITELET.	The input for options.type is not valid.
SSS_INVALID_TASK_ID	The task ID: {id} is not valid. Please refer to the documentation for a list of supported task IDs.	The base type is TASKLINK, and an invalid task ID is input for options.identifier.
SSS_INVALID_RECORD_TYPE	Type argument {type} is not a valid record or is not available in your account. Please see the documentation for a list of supported record types.	The base type is 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 base type is SUITLET, 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/https Module Script Sample](#).

```
//Add additional code
...
serverResponse.sendRedirect({
    type: 'RECORD',
    identifier: 'salesorder',
    parameters: {entity: 8}
});
...
//Add additional code
```

ServerResponse.setHeader(options)

Method Description	Method used to set the value of a response header.
Returns	Void
Supported Script Types	Server-side scripts

Governance	None
Module	N/https Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.name	string	required	The name of the header.	Version 2015 Release 2
options.value	string	required	The value used to set the header.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.
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.renderPdf(options)

Method Description	Method used to generates and renders a PDF directly to the response.
Returns	Void
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/https Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.xmlString	string	required	Content of the pdf.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax



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

```
//Add additional code
...
serverResponse.renderPDF({
    xmlString:'<?xml version="1.0"?>\n<!DOCTYPE pdf PUBLIC "-//big.faceless.org//report" "report-1.1.dtd">\n<pdf>\n<body font-size="18">\nHello World!\n</body>\n</pdf>'
});
...
//Add additional code
```

ServerResponse.setCdnCacheable(options)

Method Description	Method used to set CDN caching for a period of time.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The value of the caching duration. Set using the https.CacheDuration .	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax

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

```
//Add additional code
...
serverResponse.setCdnCacheable({
    type: https.CacheDuration.MAX
});
...
//Add additional code
```

ServerResponse.write(options)

Method Description	Method used to write information to the response.  Note: This method accepts only strings. To pass in a file, you can use ServerResponse.writeFile(options) .
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.output	string	required	The output string being written.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: argument: {param name}	A required parameter is not passed.

Error Code	Message	Thrown If
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)

Method Description	Method used to write a file to the response.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.file	file.File	required	A file.File Object that encapsulates the file to be written.	Version 2015 Release 2
options.isInline	boolean true false	optional	Determines whether the field is inline. If true, the file is inline.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.
WRONG_PARAMETER_TYPE	{param name}	The value input for options.file is not a file.File 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)

Method Description	Method used to write line information to the response.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.output	string	required	The output string being written.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.
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)

Method Description	Method used to generate a page.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
options.pageObject	serverWidget.Assistant serverWidget.Form serverWidget.List	required	A standalone page object in the form of an assistant, form or list.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax



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

```

//Add additional code
...
var myPageObj = serverWidget.createList({
    title: 'Simple List'
});

ServerResponse.writePage({
    pageObject: myPageObj
});
...

```

```
//Add additional code
```

ServerResponse.headers

Property Description	The server response headers. This property is read-only.
Type	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>
Module	N/https Module
Since	Version 2015 Release 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
...
if serverResponse.headers.ContentType === 'text/plain'
return true

log.debug({
  title: "Server Response Headers",
  details: serverResponse.headers
});
...
//Add additional code
```

https.get(options)

Method Description	Method used to send an HTTPS GET request and return the response
Returns	https.ClientResponse
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/https Module

Since	Version 2015 Release 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	Version 2015 Release 2
options.headers	object	optional	The HTTPS headers.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax

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

```
//Add additional code
...
var response = https.get({
    url: 'https://www.testwebsite.com'
});
...
//Add additional code
```

https.get.promise(options)

Method Description	Method used to send an HTTPS GET request asynchronously and return the response
	 Note: For information about the parameters and errors thrown for this method, see https.get(options) . For additional information on promises, see Promise object .
Returns	A https.ClientResponse object.
Synchronous Version	https.get(options)
Supported Script Types	All client-side scripts

Governance	10 units
Module	N/https Module
Since	Version 2015 Release 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
...
https.get.promise({
    url: 'http://www.testwebsite.com'
})
    .then(function(response){
        log.debug({
            title: 'Response',
            details: response
        });
    })
    .catch(function onRejected(reason) {
        log.debug({
            title: 'Invalid Get Request: ',
            details: reason
        });
    })
...
//Add additional code
```

https.delete(options)

Method Description	<p>Method used to send an HTTPS DELETE request and returns the response.</p> <p>⚠ Important: If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs.</p> <p>ⓘ Note: This method does not include an <code>options.body</code> parameter. Postdata is not required when the HTTPS method is a DELETE request.</p>
Returns	<code>https.ClientResponse</code>
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/https Module
Since	Version 2015 Release 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	Version 2015 Release 2
options.headers	object	optional	The HTTPS headers.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax

 Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample .
<pre>//Add additional code ... var response = https.delete({ url: 'http://www.mytestwebsite.com' }); ... //Add additional code</pre>

https.delete.promise(options)

Method Description	Method used to send an HTTP DELETE request asynchronously and return the response.  Note: For information about the parameters and errors thrown for this method, see https.delete(options) . For additional information on promises, see Promise object .
Returns	A https.ClientResponse object
Synchronous Version	https.delete(options)
Supported Script Types	All client-side scripts
Governance	10 units
Module	N/https Module
Since	Version 2015 Release 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
...
https.delete.promise({
    url: 'http://www.mytestwebsite.com'
})
    .then(function(response){
        log.debug({
            title: 'Response',
            details: response
        });
    })
    .catch(function onRejected(reason) {
        log.debug({
            title: 'Invalid Request: ',
            details: reason
        });
    })
...
//Add additional code
```

https.request(options)

Method Description	Method used to send an HTTPS request and return the response.
	<p>⚠️ Important: If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs.</p>
Returns	An https.ClientResponse Object
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/https Module
Since	Version 2015 Release 2

Parameters

ℹ️ 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 https.Method enum .	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.url	string	required	The HTTPS URL being requested	Version 2015 Release 2
options.body	string object	optional	The POST data if the method is POST.	
			<p>Note: If the method is DELETE, this body data is ignored.</p>	
options.headers	object	optional	An object containing request headers.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax



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

```
//Add additional code
...
var response = https.request({
    method: https.Method.GET,
    url: 'https://www.testwebsite.com'
});
...
//Add additional code
```

https.request.promise(options)

Method Description	Method used to send an HTTP request asynchronously and return the response.
	<p>Note: For information about the parameters and errors thrown for this method, see https.request(options). For additional information on promises, see Promise object.</p>
Returns	A https.ClientResponse object
Synchronous Version	https.request(options)

Supported Script Types	All client-side scripts
Governance	10 units
Module	N/https Module
Since	Version 2015 Release 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
...
https.request.promise({
    method: https.Method.GET,
    url: 'https://www.testwebsite.com'
})
    .then(function(response){
        log.debug({
            title: 'Response',
            details: response
        });
    })
    .catch(function onRejected(reason) {
        log.debug({
            title: 'Invalid Request: ',
            details: reason
        });
    })
...
//Add additional code
```

https.post(options)

Method Description	Method used to send an HTTPS POST request and return the response.
	 Important: If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs.
Returns	An https.ClientResponse Object
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/https Module
Since	Version 2015 Release 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	Version 2015 Release 2
options.body	string object	required	The POST data.	
options.headers	object	optional	The HTTPS headers.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax



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

```
//Add additional code
...
var response = https.post({
    url: 'https://www.testwebsite.com',
    body: mypostDataObj
});
...
//Add additional code
```

https.post.promise(options)

Method Description	Method used to send an HTTPS POST request asynchronously and return the response.
	 Note: For information about the parameters and errors thrown for this method, see https.post(options) . For additional information on promises, see Promise object .
Returns	A https.ClientResponse object
Synchronous Version	https.post(options)

Supported Script Types	All client-side scripts
Governance	10 units
Module	N/https Module
Since	Version 2015 Release 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
...
https.post.promise({
    url: 'https://www.testwebsite.com',
    body: myPostDataObj
})
    .then(function(response){
        log.debug({
            title: 'Response',
            details: response
        });
    })
    .catch(function onRejected(reason) {
        log.debug({
            title: 'Invalid Request: ',
            details: reason
        });
    })
...
//Add additional code
```

https.put(options)

Method Description	Method used to send an HTTPS PUT request and return server response.
	⚠️ Important: If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs.
Returns	An https.ClientResponse Object
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/https Module
Since	Version 2015 Release 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	Version 2015 Release 2
options.body	string object	required	The PUT data.	
options.headers	object	optional	The HTTPS headers.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	Missing a required argument: {param name}	A required parameter is not passed.

Syntax

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

```
//Add additional code
...
var response = https.put({
    url: 'https://www.testwebsite.com',
    body: myDataObj,
    headers: headerObj
});
...
//Add additional code
```

https.put.promise(options)

Method Description	Method used to send an HTTPS PUT request asynchronously and return the response.
	 Note: For information about the parameters and errors thrown for this method, see https.put(options) . For additional information on promises, see Promise object .
Returns	https.ClientResponse object
Synchronous Version	https.put(options)
Supported Script Types	All client-side scripts

Governance	10 units
Module	N/https Module
Since	Version 2015 Release 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
...
https.put.promise({
    url: 'https://www.testwebsite.com',
    body: myDataObj,
    headers: headerObj
})
    .then(function(response){
        log.debug({
            title: 'Response',
            details: response
        });
    })
    .catch(function onRejected(reason) {
        log.debug({
            title: 'Invalid Request: ',
            details: reason
        });
    })
...
//Add additional code
```

https.CacheDuration

Enum Description	Holds the string values for supported cache durations. This enum is used to set the value of the ServerResponse.setCdnCacheable(options) property.
Module	N/https Module

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

https.Encoding

Enum Description	<p>Holds the string values for supported encoding values.</p> <p>i Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Module	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](#)

https.Method

Enum Description	<p>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>
-------------------------	---

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

Values

- DELETE
- GET
- 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
```

N/log Module

Use the log module to access methods for logging script execution details.

The log methods can be accessed globally or by loading this module. Load the N/log module when you want to manually access its members, such as for testing purposes. For more information about global objects, see [SuiteScript 2.0 Global Objects and Methods](#).



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

- [N/log Module Members](#)
- [N/log Module Guidelines](#)
- [Using Log Levels](#)
- [Viewing Script Execution Logs](#)
- [log Module Script Sample](#)

N/log Module Members

Member Type	Name	Return Type / Value Type	Description
Method	log.audit(options)	Void	Logs an entry of type AUDIT to the Execution Log tab of the script deployment for the current script.
	log.debug(options)	Void	Logs an entry of type DEBUG to the Execution Log tab of the script deployment for the current script.
	log.emergency(options)	Void	Logs an entry of type EMERGENCY to the Execution Log tab of the script deployment for the current script.
	log.error(options)	Void	Logs an entry of type ERROR to the Execution Log tab of the script deployment for the current script.

N/log Module Guidelines

- NetSuite governs the amount of logging that can be done in any specific 60 minute time period. A company is allowed to make up to 100,000 log object method calls across **all** of their scripts. Script owners are notified if NetSuite detects that one script is logging excessively and automatically adjusts the log level.
- NetSuite purges system errors older than **60 days** and user-generated logs older than **30 days**. Because log persistence is not guaranteed, NetSuite recommends using custom records if you want to store script execution logs for extended periods.
- The Execution Log tab also lists notes returned by NetSuite such as error messages. For more information, see [N/error Module](#).
- 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.

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.

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.

log Module Script Sample

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

```
/** 
 * @NApiVersion 2.x
 */
require(['N/log'],
    function(myLog) {
        var myObject = {
            name: 'Jane',
            id: '123'
        };
        myLog.debug({
            title: 'hello!'
        });
        myLog.debug({
            title: 'hello!',
            details: 'world'
        });
        myLog.debug({
            title: 'myObj',
            details: myObject
        });
    });
});
```

log.audit(options)

Method Description	Logs an entry of type AUDIT to the Execution Log tab of the script deployment for the current script. This entry will not appear on the Execution Log tab if the Log Level field for the script deployment is set to ERROR or above. Use this method for scripts in production.
Returns	void
Supported Script Types	All script types
Governance	Amount of logging in any 60 minute period is limited. See N/log Module Guidelines .
Module	N/log Module
Since	Version 2016 Release 1

Parameters

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

Parameter	Type	Required / Optional	Description	Since
title	string	Optional	String to appear in the Title column on the Execution Log tab of the script deployment. Maximum length is 99 characters. If you set this value to null, an empty string (""), or omit it, the word "Untitled" appears for the log entry.	Version 2016 Release 1
details	any	Required	You can pass any value for this parameter. If the value is a JavaScript Object type, JSON.stringify(obj) is called on the object before displaying the value. NetSuite truncates any resulting string over 3999 characters.	Version 2016 Release 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)

Method Description	Logs an entry of type DEBUG 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 AUDIT or above. Use this method for scripts in development.
Returns	void
Supported Script Types	All script types
Governance	Amount of logging in any 60 minute period is limited. See N/log Module Guidelines .
Module	N/log Module
Since	Version 2016 Release 1

Parameters

Note: The options parameter is a JavaScript object.				
Parameter	Type	Required / Optional	Description	Since
title	string	Optional	String to appear in the Title column on the Execution Log tab of the script deployment. Maximum length is 99 characters. If you set this value to null, an empty string (""), or omit it, the word "Untitled" appears for the log entry.	Version 2016 Release 1
details	any	Required	You can pass any value for this parameter. If the value is a JavaScript object type, JSON.stringify(obj) is called on the object before displaying the value. NetSuite truncates any resulting string over 3999 characters.	Version 2016 Release 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)

Method Description	Logs an entry of type EMERGENCY to the Execution Log tab of the script deployment for the current script. Use this method for scripts in production.
Returns	void
Supported Script Types	All script types
Governance	Amount of logging in any 60 minute period is limited. See N/log Module Guidelines .
Module	N/log Module
Since	Version 2016 Release 1

Parameters

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

Parameter	Type	Required / Optional	Description	Since
title	string	Optional	String to appear in the Title column on the Execution Log tab of the script deployment. Maximum length is 99 characters. If you set this value to null, an empty string (''), or omit it, the word "Untitled" appears for the log entry.	Version 2016 Release 1
details	any	Required	You can pass any value for this parameter. If the value is a JavaScript Object type, JSON.stringify(obj) is called on the object before displaying the value. NetSuite truncates any resulting string over 3999 characters.	Version 2016 Release 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)

Method Description	Logs an entry of type ERROR to the Execution Log tab of the script deployment for the current script. This entry will not appear on the Execution Log tab if the Log Level field for the script deployment is set to EMERGENCY or above. Use this method for scripts in production.
Returns	void
Supported Script Types	All script types
Governance	Amount of logging in any 60 minute period is limited. See N/log Module Guidelines .
Module	N/log Module
Since	Version 2016 Release 1

Parameters

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

Parameter	Type	Required / Optional	Description	Since
title	string	Optional	String to appear in the Title column on the Execution Log tab of the script deployment. Maximum length is 99 characters. If you set this value to null, an empty string (''), or omit it, the word "Untitled" appears for the log entry.	Version 2016 Release 1
details	any	Required	You can pass any value for this parameter. If the value is a JavaScript object type, JSON.stringify(obj) is called on the object before displaying the value. NetSuite truncates any resulting string over 3999 characters.	Version 2016 Release 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/plugin Module

Load the N/plugin module to load custom plug-in implementations.

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

- [N/plugin Module Members](#)
- [N/plugin Module Script Samples](#)

N/plugin Module Members

Member Type	Name	Return Type / Value Type	Description
Method	plugin.findImplementations(options)	string[]	Returns the script IDs of custom plug-in type implementations.
Method	plugin.loadImplementation(options)	Object	Instantiates an implementation of the custom plug-in type.

N/plugin Module Script Samples

This example iterates through all implementations of a custom plug-in (assuming that the plug-in type was created with Script ID 'customscript_magic_plugin') and invokes the implementation.

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

```
/**
 * @NApiVersion 2.0
 */
require(['N/plugin'],
    function(plugin) {
        // get all implementations of the custom plugin type and iterate
        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));
}

// use default implementation (the one which is currently selected as 'active' in UI)
var pl = plugin.loadImplementation({
    type: 'customscript_magic_plugin'
});
log.debug('default impl result = ' + pl.doTheMagic(10, 20));
});
```

The following example shows how a custom plug-in is implemented:

```

/**
 * @NApiVersion 2.0
 * @NScriptType PluginTypeImpl
 */

define(function() {
    return {
        doTheMagic: function(operand1, operand2) {
            return operand1 + operand2;
        }
    };
});
```

plugin.findImplementations(options)

Method Description	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.
Returns	A string[] containing a list of custom plug-in implementation script IDs.
Supported Script Types	Server-side scripts
Governance	
Module	N/config Module
Since	Version 2016 Release 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.	Version 2016 Release 1

Parameter	Type	Required / Optional	Description	Since
options.includeDefault	boolean true false	optional	The default value is true, indicating that the default implementation should be included in the list.	Version 2016 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/plugin Module Script Samples](#).

```
//Add additional code
...
var impls = plugin.findImplementations({
    type: 'customscript_sample_plugin'
});
...
//Add additional code
```

plugin.loadImplementation(options)

Method Description	Instantiates an implementation of the custom plugin type. Returns the implementation which is currently selected in the UI (Manage Plug-ins page) when no implementation ID is explicitly provided.
Returns	An Object implementing the custom plug-in type.
Supported Script Types	Server-side scripts
Governance	
Module	N/config Module
Since	Version 2016 Release 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.	Version 2016 Release 1
options.implementation	string	optional	The script ID of the custom plug-in implementation.	Version 2016 Release 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 Samples](#).

```
//Add additional code
...
var pl = plugin.loadImplementation({
    type: 'customscript_sample_plugin'
});
...
//Add additional code
```

N/portlet Module

Load the portlet module to resize or refresh a form portlet. See [Portlet Script Type](#).

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

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

N/portlet Module Members

Member Type	Name	Return Type	Description
Method	portlet.resize	void	Resizes a form portlet immediately.
	portlet.refresh	void	Refreshes a form portlet immediately.

N/portlet Module Script Sample

The following sample shows how to create a form portlet that allows users to adjust its height and width. It creates two text fields representing the height and width of the portlet, measured in pixels. It also creates a button that runs the resize function to adjust the height and width of the portlet based on the values of the text fields.

The sample also shows how to create a button that uses the refresh function. When pressed, the portlet is updated to show the current date.

```
/**
 * @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.refresh(); }) \\" href='#{'>Refresh</a><br>";
    }
}

```

```

pts/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

Method Description	Resizes a form portlet type immediately.
Returns	Void
Supported Script Types	Client scripts
Governance	None
Module	N/portlet Module
Since	Version 2016 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/portlet Module Script Sample](#).

```

//Add additional code
...
portlet.resize();
...
//Add additional code

```

portlet.refresh

Method Description	Refreshes a form portlet type immediately.
Returns	Void
Supported Script Types	Client scripts
Governance	None
Module	N/portlet Module
Since	Version 2016 Release 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/record Module

Load the record module to work with NetSuite records. For example, to create, delete, copy, or load a record. You can also use this module to access body or sublist properties on 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](#). For help working with an instance of a custom record type, see the help topic [Custom Record](#).



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/record Module Members](#)
- [Column Object Members](#)
- [Field 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	Description
Object	record.Column	Object	Encapsulates a column of a sublist on a standard or custom record.

Member Type	Name	Return Type / Value Type	Description
	record.Field	Object	Encapsulates a body or sublist field on a standard or custom record.
	record.Record	Object	Encapsulates a NetSuite record.
	record.Sublist	Object	Encapsulates a sublist on a standard or custom record.
Method	record.attach(options)	void	Attaches a record to another record.
	record.attach.promise(options)	Promise	Attaches a record asynchronously to another record.
	record.copy(options)	record.Record	Creates a new record by copying an existing record in NetSuite.
	record.copy.promise(options)	Promise	Creates a new record asynchronously by copying an existing record in NetSuite.
	record.create(options)	record.Record	Creates a new record.
	record.create.promise(options)	Promise	Creates a new record asynchronously.
	record.delete(options)	number	Deletes a record.
	record.delete.promise(options)	Promise	Deletes a record asynchronously.
	record.detach(options)	void	Detaches a record from another record.
	record.detach.promise(options)	Promise	Detaches a record from another record asynchronously.
	record.load(options)	record.Record	Loads an existing record.
	record.load.promise(options)	Promise	Loads an existing record asynchronously.
	record.submitFields(options)	number	Updates and submits one or more body fields on an existing record in NetSuite, and returns the internal ID of the parent record.
	record.submitFields.promise(options)	Promise	Updates and submits one or more body fields asynchronously on an existing record in NetSuite, and returns the internal ID of the parent record.

Member Type	Name	Return Type / Value Type	Description
	record.transform(options)	record.Record	Transforms a record from one type into another, using data from an existing record.
	record.transform.promise(options)	Promise	Transforms a record from one type into another asynchronously, using data from an existing record.
Enum	record.Type	enum	Enumeration that holds the string values for supported record types.

Column Object Members

Member Type	Name	Return Type / Value Type	Description
Property	Column.id	string (read-only)	Returns the internal ID of the column.
	Column.label	string (read-only)	Returns the UI label for the column.
	Column.sublistId	string (read-only)	Returns the internal ID of the standard or custom sublist that contains the column.
	Column.type	string (read-only)	Returns the column type.

Field Object Members

Member Type	Name	Return Type / Value Type	Description
Method	Field.getSelectOptions(options)	array	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)	Returns the UI label for a standard or custom field body or sublist field.
	Field.id	string (read-only)	Returns the internal ID of a standard or custom body or sublist field.
	Field.type	string (read-only)	Returns the type of a body or sublist field.
	Field.isMandatory	boolean true false	Returns true if the standard or custom field is mandatory on the record form, or false otherwise.

Member Type	Name	Return Type / Value Type	Description
	Field.sublistId	string (read-only)	Returns the ID of the sublist associated with the specified sublist field.

Record Object Members

The following members are called on the `record.Record` object.

Member Type	Name	Return Type / Value Type	Description
Method	Record.cancelLine(options)	record.Record	Cancels the currently selected line on a sublist.
	Record.commitLine(options)	record.Record	Commits the currently selected line on a sublist.
	Record.findMatrixSublistLineWithValue(options)	number	Returns the line number of the first instance where a specified value is found in a specified column of the matrix.
	Record.findSublistLineWithValue(options)	number	Returns the line number for the first occurrence of a field value in a sublist.
	Record.getCurrentMatrixSublistValue(options)	number Date string array boolean true false	Gets the value for the currently selected line in the matrix.
	Record.getCurrentSublistField(options)	record.Field	Returns a field object from a sublist.
	Record.getCurrentSublistIndex(options)	number	Returns the line number of the currently selected line.
	Record.getCurrentSublistSubrecord(options)	record.Record	Gets the subrecord for the associated sublist field on the current line.
	Record.getCurrentSublistText(options)	string	Returns a text representation of the field value in the currently selected line.
	Record.getCurrentSublistValue(options)	number Date string array boolean true false	Returns the value of a sublist field on the currently selected sublist line.
	Record.getField(options)	record.Field	Returns a field object from a record.
	Record.getFields()	string[]	Returns the body field names (internal ids) of all the fields in the record, including machine header field and matrix header fields.

Member Type	Name	Return Type / Value Type	Description
	Record.getLineCount(options)	number	Returns the number of lines in a sublist.
	Record.getMatrixHeaderCount(options)	number	Returns the number of columns for the specified matrix.
	Record.getMatrixHeaderFieldId(options)	record.Field	Gets the field for the specified header in the matrix.
	Record.getMatrixHeaderValue(options)	number Date string array boolean true false	Gets the value for the associated header in the matrix.
	Record.getMatrixSublistFieldId(options)	record.Field	Gets the field for the specified sublist in the matrix.
	Record.getMatrixSublistValue(options)	number Date string array boolean true false	Gets the value for the associated field in the matrix.
	Record.getSublist(options)	record.Sublist	Returns the specified sublist.
	Record.getSublists()	string[]	Returns all the names of all the sublists.
	Record.getSublistField(options)	record.Field	Returns a field object from a sublist.
	Record.getSublistFields(options)	string[]	Returns all the field names in a sublist.
	Record.getSublistSubrecord(options)	record.Record	Gets the subrecord associated with a sublist field. (standard mode only)
	Record.getSublistText(options)	string	Returns the value of a sublist field in a text representation.
	Record.getSublistValue(options)	number Date string array boolean true false	Returns the value of a sublist field.
	Record.getSubrecord(options)	record.Record	Gets the subrecord for the associated field.
	Record.getText(options)	string	Returns the text representation of a field value.
	Record.getValue(options)	number Date string array boolean true false	Returns the value of a field.

Member Type	Name	Return Type / Value Type	Description
	Record.hasCurrentSublistSubrecord(options)	boolean true false	Returns a value indicating whether the associated sublist field has a subrecord on the current line.
	Record.hasSublistSubrecord(options)	boolean true false	Returns a value indicating whether the associated sublist field contains a subrecord.
	Record.hasSubrecord(options)	boolean true false	Returns a value indicating whether the field contains a subrecord.
	Record.insertLine(options)	record.Record	Inserts a sublist line.
	Record.removeCurrentSublistSubrecord(options)	record.Record	Removes the subrecord for the associated sublist field on the current line.
	Record.removeLine(options)	record.Record	Removes a sublist line.
	Record.removeSublistSubrecord(options)	record.Record	Removes the subrecord for the associated sublist field. (standard mode only)
	Record.removeSubrecord(options)	record.Record	Removes the subrecord for the associated field.
	Record.save(options)	number	Submits a new record or saves edits to an existing record.
	Record.save.promise(options)	number	Submits a new record asynchronously or saves edits to an existing record asynchronously.
	Record.selectLine(options)	record.Record	Selects an existing line in a sublist.
	Record.selectNewLine(options)	record.Record	Selects a new line at the end of a sublist.
	Record.setCurrentMatrixSublistValue(options)	record.Record	Sets the value for the line currently selected in the matrix.
	Record.setCurrentSublistText(options)	record.Record	Sets the value for the field in the currently selected line by a text representation.
	Record.setCurrentSublistValue(options)	record.Record	Sets the value for the field in the currently selected line.
	Record.setMatrixHeaderValue(options)	record.Record	Sets the value for the associated header in the matrix.

Member Type	Name	Return Type / Value Type	Description
	Record.setMatrixSublistValue(options)	record.Record	Sets the value for the associated field in the matrix.
	Record.setSublistText(options)	record.Record	Sets the value of a sublist field by a text representation. (standard mode only)
	Record.setSublistValue(options)	record.Record	Sets the value of a sublist field. (standard mode only)
	Record.setText(options)	record.Record	Sets the value of the field by a text representation.
	Record.setValue(options)	record.Record	Sets the value of a field.
Property	Record.id	number (read-only)	The internal ID of a specific record.
	Record.isDynamic	boolean (read-only)	Indicates whether the record is in dynamic or standard mode.
	Record.type	string (read-only)	The record type.

Sublist Object Members

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

N/record Module Script Samples

The following script samples demonstrate how to use the record module.

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



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.

The following example shows how to create and save a contact record.

```
/**
 * @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();
});

```

The following example shows how to create and save a contact record using Promise methods.

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

```

```

        value: '1'
    });
    for ( var key in nameData) {
        if (nameData.hasOwnProperty(key)) {
            objRecord.setValue({
                fieldId: key,
                value: nameData[key]
            });
        }
    }
    var recordId = objRecord.save({
        enableSourcing: false,
        ignoreMandatoryFields: false
    });
}, function(e) {
    log.error('Unable to create contact', e.name);
});
}
createAndSaveContactRecordWithPromise();
});

```

The following example shows how to access sublists and a subrecord from a record. This example requires the Advanced Number Inventory Management feature.

 **Note:** For additional script samples that include subrecords, see [Scripting Subrecords](#).

```

/**
 * @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'
    });
}
createPurchaseOrder();
});

```

The following example shows how to access sublists and a subrecord from a record using Promise methods. This example requires the Advanced Number Inventory Management feature.

```

/**
 * @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: 'myinventoryNumber'
            });
            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'
            });
        })
    }
}
);

```

```

        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'
    });
}, function(err) {
    log.error('Unable to create purchase order!', err.name);
});
}
createPurchaseOrder();
});

```

N/record Default Values

You can use SuiteScript 2.0 to specify record initialization parameters that default when creating, copying, loading, and transforming records. To enable this behavior, use the optional `defaultValues` parameter in the following APIs:

- `record.create(options)`
- `record.copy(options)`
- `record.transform(options)`
- `record.load(options)`

The following table lists initialization types that are available to certain SuiteScript-supported records and the values they can contain.

Record	Initialization Type	Values
All SuiteScript-supported records that support form customization.	customform	<customformid>

Record	Initialization Type	Values
Assembly Build	assemblyitem	<assemblyitemid>
Cash Refund	entity	<entityid>
Cash Sale	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
Tax Group	nexuscountry	<countrycode> See Country Codes Used for Initialization Parameters .
Tax Type	country	<countrycode> See Country Codes Used for Initialization Parameters .
Topic	parenttopic	<parenttopicid>
Vendor Bill	entity	<entityid>
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
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

Country Code	Country Name
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
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

Country Code	Country Name
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
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

Country Code	Country Name
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
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

Country Code	Country Name
MH	Marshall Islands
MK	Macedonia
ML	Mali
MM	Myanmar
MN	Mongolia
MO	Macau
MP	Northern Mariana Islands
MQ	Martinique
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

Country Code	Country Name
PL	Poland
PM	St. Pierre and Miquelon
PN	Pitcairn Island
PR	Puerto Rico
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

Country Code	Country Name
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
YT	Mayotte
ZA	South Africa
ZM	Zambia
ZW	Zimbabwe

record.Column

Object Description	Encapsulates a column of a sublist on a standard or custom record. For a complete list of this object's properties, see Column Object Members .
Supported Script Types	Client and server-side scripts
Module	N/record Module
Since	Version 2015 Release 2

Syntax

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

Column.id

Property Description	Returns the internal ID of the column.
Type	string (read-only)
Module	N/record Module
Since	Version 2015 Release 2

Syntax

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

Column.label

Property Description	Returns the internal ID of the column.
-----------------------------	--

Type	string (read-only)
Module	N/record Module
Since	Version 2015 Release 2

Syntax



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

Column.sublistId

Property Description	Returns the internal ID of the standard or custom sublist that contains the column.
Type	string (read-only)
Module	N/record Module
Since	Version 2015 Release 2

Syntax



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

Column.type

Property Description	Returns the column type.
Type	string (read-only)
Module	N/record Module
Since	Version 2015 Release 2

Syntax



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

record.Field

Object Description	Encapsulates a body or sublist field on a standard or custom record. Use the following methods to access the Field object: <ul style="list-style-type: none">▪ Record.getField(options)▪ Record.getSublistField(options)▪ Record.getCurrentSublistField(options)▪ CurrentRecord.getField(options)▪ CurrentRecord.getSublistField(options) For a complete list of this object's methods and properties, see Field Object Members .
Supported Script Types	Client and server-side scripts
Module	N/record Module
Since	Version 2015 Release 2

Syntax



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: 'entity'
});
...
//Add additional code
```

Field.getSelectOptions(options)

Method Description	Returns an array of available options on a standard or custom select, multi-select, or radio field as key-value pairs. Only the first 1,000 available options are returned.
---------------------------	---

	<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 <code>Type Error</code> if the field is not a select field.</p> <p>Important: You can only use this method on a record in dynamic mode. For additional information on dynamic mode, see <code>record.Record</code>.</p>
Returns	array
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 2

Parameters

<p>Note: The options parameter is a JavaScript object.</p>				
Parameter	Type	Required / Optional	Description	Since
options.filter	string	Required	<p>The search string to filter the select options that are returned.</p> <p>Note: Filter values are case insensitive.</p>	Version 2015 Release 2
options.operator	string	Required	<p>The following operators are supported:</p> <ul style="list-style-type: none"> ■ <code>contains</code> (default) ■ <code>is</code> ■ <code>startswith</code> 	Version 2015 Release 2

Syntax

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 options = objField.getSelectOptions({
    filter : 'C',
    operator : 'startswith'
});
...
```

```
//Add additional code
```

Field.label

Property Description	Returns the UI label for a standard or custom field body or sublist field.
Type	string (read-only)
Module	N/record Module
Since	Version 2015 Release 2

Syntax

 **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 = objField.label;
...
//Add additional code
```

Field.id

Property Description	Returns the internal ID of a standard or custom body or sublist field.
Type	string (read-only)
Module	N/record Module
Since	Version 2015 Release 2

Syntax

 **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 = objField.id;
...
//Add additional code
```

Field.type

Property Description	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.
-----------------------------	--

Type	string (read-only)
Module	N/record Module
Since	Version 2015 Release 2

Syntax

 **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 type = objField.type;
...
//Add additional code
```

Field.isMandatory

Property Description	Returns <code>true</code> if the standard or custom field is mandatory on the record form, or <code>false</code> otherwise.
Type	boolean <code>true</code> <code>false</code>
Module	N/record Module
Since	Version 2015 Release 2

Syntax

 **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 (objField.isMandatory) {
    ...
}
...
//Add additional code
```

Field.sublistId

Property Description	Returns the sublist ID for the specified sublist field.
Type	string (read-only)
Module	N/record Module
Since	Version 2015 Release 2

Syntax



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 myId = field sublistId;
...
//Add additional code
```

record.Record

Load the record module when you want to work with NetSuite records.

Object Description	Encapsulates a NetSuite record. There are two modes you can operate in when you create, copy, load, or transform a record with SuiteScript 2.0: standard mode and dynamic mode. <ul style="list-style-type: none"> ■ When a SuiteScript 2.0 script creates, copies, loads, or transforms a record in standard mode, the record's body fields and sublist line items are not sourced, calculated, and validated until the record is saved (submitted) with Record.save(options). When you work with a record in standard mode, you do not need to set values in any particular order. After submitting the record, NetSuite processes the record's body fields and sublist line items in the correct order, regardless of the organization of your script. ■ When a SuiteScript 2.0 script creates, copies, loads, or transforms a record in dynamic mode, the record's body fields and sublist line items are sourced, calculated, and validated in real-time. A record in dynamic mode emulates the behavior of a record in the UI. When you work with a record in dynamic mode, it is important that you set values in the same order you would within the UI. If you fail to do this, your results may not be accurate. The record.create(options) , record.copy(options) , record.load(options) , and record.transform(options) methods work in standard mode by default. If you want these methods to work in dynamic mode, you must pass in a specific argument. See the help topic for the applicable method for more information. For a complete list of this object's methods and properties, see Record Object Members .
Supported Script Types	Client and server-side scripts
Module	N/record Module
Since	Version 2015 Release 2

Syntax



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)

Method Description	Cancels the currently selected line on a sublist.
Returns	The record.Record object that called the method.
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 2

Parameters

 Note: 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> <th>Since</th> </tr> </thead> <tbody> <tr> <td>options.sublistId</td> <td>string</td> <td>required</td> <td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td> <td>Version 2015 Release 2</td> </tr> </tbody> </table>	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 Using the SuiteScript Records Browser .	Version 2015 Release 2
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 Using the SuiteScript Records Browser .	Version 2015 Release 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)

Method Description	Commits the currently selected line on a sublist.
Returns	The record.Record object that called the method.
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 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.commitLine({
    sublistId: 'item'
});
...
//Add additional code

```

Record.findMatrixSublistLineWithValue(options)

Method Description	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.
Returns	number
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	2016.2
options.fieldId	string	required	The ID of the matrix field.	2016.2
options.value	number	required	The value to search for.	2016.2
options.column	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
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 ...
var lineNumber = objRecord.findMatrixSublistLineWithValue({
    sublistId: 'item'
});
...
//Add additional code
```

Record.findSublistLineWithValue(options)

Method Description	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.
Returns	A line number as a number, or -1 if not found.
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	Version 2015 Release 2
options.value	number Date string	optional	The value to search for.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
	array boolean true false			

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)

Method Description	Gets the value for the currently selected line in the matrix.
Returns	number Date string array boolean true false
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser .	
options.fieldId	string	required	The internal ID of the matrix field.	Version 2015 Release 2
options.column	number	required	The column number for the matrix field. Note that column indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 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 matrixValue = objRecord.getCurrentMatrixSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 12
});
...
//Add additional code
```

Record.getCurrentSublistField(options)

Method Description	Returns metadata about a sublist field.  Important: You can only use this method on a record in dynamic mode. For additional information on dynamic mode, see record.Record .
Returns	record.Field
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module

Since	Version 2016 Release 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.	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 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)

Method Description	Returns the line number of the currently selected line. Note that line indexing begins at 0 with SuiteScript 2.0.
Returns	number
Supported Script Types	Client and server-side scripts

Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 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 currIndex = objRecord.getCurrentSublistIndex({
    sublistId: 'item'
});
...
//Add additional code
```

Record.getCurrentSublistSubrecord(options)

Method Description	Gets the subrecord for the associated sublist field on the current line.
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None

Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	Version 2015 Release 2

Syntax



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)

Method Description	Returns a text representation of the field value in the currently selected line.
Returns	string
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	Version 2015 Release 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)

Method Description	Returns the value of a sublist field on the currently selected sublist line.
Returns	number Date string array boolean true false
Supported Script Types	Client and server-side scripts
Governance	None

Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	Version 2015 Release 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)

Method Description	Returns a field object from a record.
--------------------	---------------------------------------

Returns	record.Field
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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.	Version 2015 Release 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()

Method Description	Returns the body field names (internal ids) of all the fields in the record, including machine header field and matrix header fields.
Returns	string[]
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module

Since	Version 2015 Release 2
-------	------------------------

Syntax



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)

Method Description	Returns the number of lines in a sublist.
Returns	number
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2

Syntax



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.getMatrixHeaderCount(options)

Method Description	Returns the number of columns for the specified matrix.
Returns	number
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of the matrix field.	Version 2015 Release 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)

Method Description	Gets the field for the specified header in the matrix.
Returns	record.Field
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of the matrix field.	Version 2015 Release 2
options.column	number	required	The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 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.getMatrixHeaderField({
    sublistId: 'item',
    fieldId: 'item',
    column: 12
});
...
//Add additional code

```

Record.getMatrixHeaderValue(options)

Method Description	Gets the value for the associated header in the matrix.
Returns	number Date string array boolean true false
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 that contains the matrix. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of the matrix field.	Version 2015 Release 2
options.column	number	required	The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 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.getMatrixHeaderValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 12
});
...
//Add additional code
```

Record.getMatrixSublistField(options)

Method Description	Gets the field for the specified sublist in the matrix.
Returns	record.Field
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of the matrix field.	Version 2015 Release 2
options.column	number	required	The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 2
options.line	number	required	The line number for the field. Note that line	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			indexing begins at 0 with SuiteScript 2.0.	

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.getMatrixSublistField({
    sublistId: 'item',
    fieldId: 'item',
    column: 12,
    line: 3
});
...
//Add additional code
```

Record.getMatrixSublistValue(options)

Method Description	Gets the value for the associated field in the matrix.
Returns	number Date string array boolean true false
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 that contains the matrix. This value is displayed in the Records Browser.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			For more information, see the help topic Using the SuiteScript Records Browser .	
options.fieldId	string	required	The internal ID of the matrix field.	Version 2015 Release 2
options.column	number	required	The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 2
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 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)

Method Description	Returns the specified sublist.
Returns	<code>record.Sublist</code>
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2

Syntax



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()

Method Description	Returns all the names of all the sublists.
Returns	string[]
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 2

Syntax



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)

Method Description	Returns a field object from a sublist.
Returns	record.Field
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	Version 2015 Release 2
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 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 objField = objRecord.getSublistField({
```

```

        sublistId: 'item',
        fieldId: 'item',
        line: 3
    });
...
//Add additional code

```

Record.getSublistFields(options)

Method Description	Returns all the field names in a sublist.
Returns	string[]
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 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 field = objRecord.getSublistFields({
    sublistId: 'item'
});

```

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

Record.getSublistSubrecord(options)

Method Description	Gets the sublist associated with a sublist field. (standard mode only) When working in dynamic mode, get a sublist subrecord using the following methods: <ol style="list-style-type: none">1. Record.selectLine(options)2. Record.hasCurrentSublistSubrecord(options)3. Record.getCurrentSublistSubrecord(options)
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 2

Parameters

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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	Version 2015 Release 2
options.line	number	required	The line number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 2

Syntax

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)

Method Description	Returns the value of a sublist field in a text representation.
Returns	string
Supported Script Types	Client and server-side scripts. Limitations exist on how this method can be used in standard (deferredDynamic) mode. For details, refer to the description of the SSS_INVALID_API_USAGE error code in the Errors table. In dynamic mode, you can use getSublistText() without limitation.
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	Version 2015 Release 2
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 2

Errors

Error Code	Thrown If
SSS_INVALID_API_USAGE	Invoked in certain cases when deferredDynamic mode is being used.

Error Code	Thrown If
	<p>For example, for a script that works with a remote record, this error can be invoked in both of the following situations:</p> <ul style="list-style-type: none"> ■ The record object was created by record.copy(), record.create(), or record.transform(), and the script attempts to use getSublistText() without first using setSublistText() for the same field. ■ The record object was created by record.load(), and the script uses setSublistValue() on a field before using getSublistText() for the same field. <p>This guidance also affects user event scripts that instantiate records by using the newRecord or oldRecord object provided by the script context. These records always use deferredDynamic mode. For that reason, this error appears in both of the following situations:</p> <ul style="list-style-type: none"> ■ When a user event script executes on a record that is being newly created, and the script attempts to use getSublistText() without first using setSublistText() for the same field. ■ When a user event script executes on an existing record, and the script uses setSublistValue() on a field before using getSublistText() for the same field.
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)

Method Description	Returns the value of a sublist field.
Returns	number Date string array boolean true false
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module

Since

Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	Version 2015 Release 2
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 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)

Method Description	Gets the subrecord for the associated field.
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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.	Version 2015 Release 2

Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
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)

Method Description	Returns the text representation of a field value.
Returns	string
Supported Script Types	<p>Client and server-side scripts. Limitations exist on how this method can be used in standard (deferredDynamic) mode. For details, refer to the description of the SSS_INVALID_API_USAGE error code in the Errors table. In dynamic mode, you can use getText() without limitation.</p>
Governance	None
Module	N/record Module
Since	Version 2015 Release 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.	Version 2015 Release 2

Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_API_USAGE	<p>Invoked in certain cases when deferredDynamic mode is being used. For example, for a script that works with a remote record, this error can be invoked in both of the following situations:</p> <ul style="list-style-type: none"> ■ The record object was created by record.copy(), record.create(), or record.transform(), and the script attempts to use getText() without first using setText() for the same field. ■ The record object was created by record.load(), and the script uses setValue() on a field before using getText() for the same field. <p>This guidance also affects user event scripts that instantiate records by using the newRecord or oldRecord object provided by the script context. These records always use deferredDynamic mode. For that reason, this error appears in both of the following situations:</p> <ul style="list-style-type: none"> ■ When a user event script executes on a record that is being newly created, and the script attempts to use getText() without first using setText() for the same field. ■ When a user event script executes on an existing record, and the script uses setValue() on a field before using getText() for 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 fieldidname = objRecord.getText({
    fieldId: 'item'
});
...
//Add additional code
```

Record.getValue(options)

Method Description	Returns the value of a field.
Returns	number Date string array boolean true false
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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.	Version 2015 Release 2

Errors

Error Code	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is missing or undefined.
SSS_INVALID_API_USAGE	Invoked prior to using setValue in standard record mode.

Syntax



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)

Method Description	Returns a value indicating whether the associated sublist field has a subrecord on the current line.
	 Important: You can only use this method on a record in dynamic mode. For additional information on dynamic mode, see record.Record .
Returns	boolean true false
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of a subrecord.	Version 2015 Release 2

Syntax

 **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)

Method Description	Returns a value indicating whether the associated sublist field contains a subrecord.
Returns	boolean true false
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of a subrecord.	Version 2015 Release 2
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 2

Syntax

⚠ 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)

Method Description	Returns a value indicating whether the field contains a subrecord.
Returns	boolean true false
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 field that may contain a subrecord.	Version 2015 Release 2

Syntax



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)

Method Description	Inserts a sublist line.
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.line	number	required	The line number to insert. Note that line indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 2
options.ignoreRecalc	boolean true false	optional	If set to true, scripting recalculation is ignored. The default value is false.	Version 2015 Release 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 <code>insertLine()</code> , see N/record Module Script Samples .

```
//Add additional code
...
objRecord.insertLine({
    sublistId: 'item',
    line: 3,
    ignoreRecalc: true
});
...
//Add additional code
```

For script examples that use other N/record methods, see [N/record Module Script Samples](#).

Record.removeCurrentSublistSubrecord(options)

Method Description	Removes the subrecord for the associated sublist field on the current line.
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	Version 2015 Release 2

Syntax

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)

Method Description	Removes a sublist line.
---------------------------	-------------------------

Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.line	number	required	The line number of the sublist to remove. Note that line indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 2
options.ignoreRecalc	boolean true false	optional	If set to true, scripting recalculation is ignored. The default value is false.	Version 2015 Release 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.removeLine({
```

```

    sublistId: 'item',
    line: 3,
    ignoreRecalc: true
});
...
//Add additional code

```

Record.removeSublistSubrecord(options)

Method Description	Removes the subrecord for the associated sublist field. (standard mode only) When working in dynamic mode, remove a sublist subrecord using the following methods: <ol style="list-style-type: none">1. Record.selectLine(options)2. Record.hasCurrentSublistSubrecord(options)3. Record.removeCurrentSublistSubrecord(options)4. Record.commitLine(options)
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	Version 2015 Release 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.	Version 2015 Release 2

Syntax



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)

Method Description	Removes the subrecord for the associated field.
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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.	Version 2015 Release 2

Syntax



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)

Method Description	Submits a new record or saves edits to an existing record. When working with records in standard mode, you must submit and then load the record to obtain sourced, validated, and calculated field values.
	<p>Note: This method has an asynchronous counterpart you can use with client scripts. See Record.save.promise(options).</p>
Returns	A number representing the internal ID of the new or updated record.
Supported Script Types	Client and server-side scripts
Governance	Transaction records: 20 usage units Custom records: 4 usage units All other records: 10 usage units
Module	N/record Module
Since	Version 2015 Release 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. If set to <code>true</code>, sources dependent field information for empty fields. Defaults to <code>false</code> – dependent field values are not sourced.</p> <p>Important: This parameter applies to records in standard mode only. When working with records in dynamic mode, field values are always sourced and the value you provide for <code>enableSourcing</code> is ignored.</p>	Version 2015 Release 2
options.ignoreMandatoryFields	boolean <code>true</code> <code>false</code>	optional	Disables mandatory field validation for this save operation. If set to <code>true</code> , all standard and custom fields that were made mandatory through customization are ignored. All fields that were	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<p>made mandatory through company preferences are also ignored. By default, this parameter is false.</p> <p>Important: Use the ignoreMandatoryFields argument with caution. This argument should be used mostly with Scheduled scripts, rather than User Event scripts. This ensures that UI users do not bypass the business logic enforced through form customization.</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/record Module Script Samples](#).

```
//Add additional code
...
var recordId = objRecord.save({
    enableSourcing: true,
    ignoreMandatoryFields: true
});
...
//Add additional code
```

Record.save.promise(options)

Method Description	Submits a new record asynchronously or saves edits to an existing record asynchronously.
	<p>Note: For information about the parameters and errors thrown for this method, see Record.save(options). For more information on promises, see Promise object.</p>
Returns	Promise
Supported Script Types	Client-side scripts
Governance	Transaction records: 20 usage units Custom records: 4 usage units All other records: 10 usage units
Module	N/record Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.enableSourcing	boolean true false	optional	<p>Enables sourcing during the record update. If set to <code>true</code>, sources dependent field information for empty fields. Defaults to <code>false</code> – dependent field values are not sourced.</p> <p> Important: This parameter applies to records in standard mode only. When working with records in dynamic mode, field values are always sourced and the value you provide for <code>enableSourcing</code> is ignored.</p>	Version 2015 Release 2
options.ignoreMandatoryFields	boolean true false	optional	<p>Disables mandatory field validation for this save operation. If set to <code>true</code>, all standard and custom fields that were made mandatory through customization are ignored. All fields that were made mandatory through company preferences are also ignored. By default, this parameter is <code>false</code>.</p> <p> Important: 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.</p>	Version 2015 Release 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 recordId = objRecord.save.promise({
    enableSourcing: true,
    ignoreMandatoryFields: true
});
...
//Add additional code

```

Record.selectLine(options)

Method Description	Selects an existing line in a sublist.
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.line	number	required	The line number to select in the sublist. Note that line indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 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 lineNum = objRecord.selectLine({
    sublistId: 'item',
    line: 3
});
...
//Add additional code
```

Record.selectNewLine(options)

Method Description	Selects a new line at the end of a sublist.
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 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.selectNewLine({
    sublistId: 'item'
});
...
//Add additional code
```

Record.setCurrentMatrixSublistValue(options)

Method Description	Sets the value for the line currently selected in the matrix. This method is not available for standard records.
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of the matrix field.	Version 2015 Release 2
options.column	number	required	The column number for the field. Note that column indexing	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			begins at 0 with SuiteScript 2.0.	
options.value	number Date string array boolean true false	required	<p>The value to set the field to. The value type must correspond to the field type being set. For example:</p> <ul style="list-style-type: none"> ■ Text, Radio and Select fields accept string values. ■ Checkbox fields accept Boolean values. ■ Date and DateTime fields accept Date values. ■ Integer, Float, Currency and Percent fields accept number values. 	Version 2015 Release 2
options.ignoreFieldChange	boolean true false	optional	If set to true, the field change and slaving event is ignored. By default, this value is false.	Version 2015 Release 2
options.fireSlavingSync	boolean true false	optional	Indicates whether to perform slaving synchronously. By default, this value is false.	Version 2015 Release 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,
    fireSlavingSync: true
});
...
//Add additional code

```

Record.setCurrentSublistText(options)

Method Description	Sets the value for the field in the currently selected line by a text representation.
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	Version 2015 Release 2
options.text	string	required	The text to set the value to.	Version 2015 Release 2
options.ignoreFieldChange	boolean true false	optional	If set to <code>true</code> , the field change and slaving event is ignored. By default, this value is <code>false</code> .	Version 2015 Release 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

 **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.setCurrentSublistText({
    sublistId: 'item',
    fieldId: 'item',
    text: 'value',
    ignoreFieldChange: true
});
...
//Add additional code
```

Record.setCurrentSublistValue(options)

Method Description	Sets the value for the field in the currently selected line.
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser .	
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	Version 2015 Release 2
options.value	number Date string array boolean true false	required	<p>The value to set the field to. The value type must correspond to the field type being set. For example:</p> <ul style="list-style-type: none"> ■ Text, Radio and Select fields accept string values. ■ Checkbox fields accept Boolean values. ■ Date and DateTime fields accept Date values. ■ Integer, Float, Currency and Percent fields accept number values. 	Version 2015 Release 2
options.ignoreFieldChange	boolean true false	optional	If set to <code>true</code> , the field change and slaving event is ignored. By default, this value is <code>false</code> .	Version 2015 Release 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)

Method Description	Sets the value for the associated header in the matrix.
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of the matrix field.	Version 2015 Release 2
options.column	number	required	The column number for the field. Note that column indexing	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			begins at 0 with SuiteScript 2.0.	
options.value	number Date string array boolean true false	required	<p>The value to set the field to. The value type must correspond to the field type being set. For example:</p> <ul style="list-style-type: none"> ■ Text, Radio and Select fields accept string values. ■ Checkbox fields accept Boolean values. ■ Date and DateTime fields accept Date values. ■ Integer, Float, Currency and Percent fields accept number values. 	Version 2015 Release 2
options.ignoreFieldChange	boolean true false	optional	If set to true, the field change and slaving event is ignored. By default, this value is false.	Version 2015 Release 2
options.fireSlavingSync	boolean true false	optional	Indicates whether to perform slaving synchronously. By default, this value is false.	Version 2015 Release 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.setMatrixHeaderValue({
    sublistId: 'item',
```

```

        fieldId: 'item',
        column: 3,
        value: false,
        ignoreFieldChange: true,
        fireSlavingSync: true
    });
    ...
//Add additional code

```

Record.setMatrixSublistValue(options)

Method Description	Sets the value for the associated field in the matrix.
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 that contains the matrix. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of the matrix field.	Version 2015 Release 2
options.column	number	required	The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 2
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 2
options.value	number Date string array	required	The value to set the field to. The value type must correspond to the field	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
	boolean true false		<p>type being set. For example:</p> <ul style="list-style-type: none"> ■ Text, Radio and Select fields accept string values. ■ Checkbox fields accept Boolean values. ■ Date and DateTime fields accept Date values. ■ Integer, Float, Currency and Percent fields accept number values. 	

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)

Method Description	<p>Sets the value of a sublist field by a text representation. (standard mode only)</p> <p>When working in dynamic mode, set a sublist field text using the following methods:</p> <ol style="list-style-type: none"> 1. Record.selectLine(options)
--------------------	--

	2. Record.setCurrentSublistText(options) 3. Record.commitLine(options)
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	Version 2015 Release 2
options.line	number	required	The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 2
options.text	string	required	The text to set the value to.	Version 2015 Release 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.setSublistText({
    sublistId: 'item',
    fieldId: 'item',
    line: 3,
    text: 'value'
});
...
//Add additional code

```

Record.setSublistValue(options)

Method Description	Sets the value of a sublist field. (standard mode only) When working in dynamic mode, set a sublist field value using the following methods: 1. Record.selectLine(options) 2. Record.setCurrentSublistValue(options) 3. Record.commitLine(options)
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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 Using the SuiteScript Records Browser .	Version 2015 Release 2
options.fieldId	string	required	The internal ID of a standard or custom sublist field.	Version 2015 Release 2
options.line	number	required	The line number of the sublist. Note that line indexing begins at 0 with SuiteScript 2.0.	Version 2015 Release 2
options.value	number Date string	required	The value to set the sublist field to.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
	array boolean true false		<p>The value type must correspond to the field type being set. For example:</p> <ul style="list-style-type: none"> ■ Text, Radio and Select fields accept string values. ■ Checkbox fields accept Boolean values. ■ Date and DateTime fields accept Date values. ■ Integer, Float, Currency and Percent fields accept number values. 	

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.
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.setSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    line: 3,
    value: true
});
...
//Add additional code
```

Record.setText(options)

Method Description	Sets the value of the field by a text representation.
--------------------	---

Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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.	Version 2015 Release 2
options.text	string array	required	<p>The text or texts to change the field value to.</p> <ul style="list-style-type: none"> ■ If the field type is multiselect: <ul style="list-style-type: none"> □ This parameter accepts an array of string values. □ This parameter accepts a null value. Passing in null deselects all currently selected values. ■ If the field type is not multiselect, this parameter accepts only a single string value. 	Version 2015 Release 2
options.ignoreFieldChange	boolean true false	optional	If set to <code>true</code> , the field change and slaving event is ignored. By default, this value is <code>false</code> .	Version 2015 Release 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)

Method Description	Sets the value of a field.
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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.	Version 2015 Release 2
options.value	number Date string array boolean true false	required	<p>The value to set the field to. The value type must correspond to the field type being set. For example:</p> <ul style="list-style-type: none"> ■ Text, Radio, Select and Multi-Select fields accept string values. ■ Checkbox fields accept Boolean values. 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> ■ Date and DateTime fields accept Date values. ■ Integer, Float, Currency and Percent fields accept number values. 	
options.ignoreFieldChange	boolean true false	optional	If set to <code>true</code> , the field change and slaving event is ignored. By default, this value is <code>false</code> .	Version 2015 Release 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.setValue({
    fieldId: 'item',
    value: true,
    ignoreFieldChange: true
});
...
//Add additional code
```

Record.id

Property Description	The internal ID of a specific record.
Type	number (read-only)
Module	N/record Module
Since	Version 2015 Release 2

Syntax



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 = record.id;
...
//Add additional code
```

Record.isDynamic

Property Description	<p>Indicates whether the record is in dynamic or standard mode.</p> <ul style="list-style-type: none"> ■ 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"> □ 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 <code>Record.save(options)</code>. When you work with a record in standard mode, you do not need to set values in any particular order. After submitting the record, NetSuite processes the record's body fields and sublist line items in the correct order, regardless of the organization of your script. □ When a SuiteScript 2.0 script creates, copies, loads, or transforms a record in dynamic mode, the record's body fields and sublist line items are sourced, calculated, and validated in real-time. A record in dynamic mode emulates the behavior of a record in the UI. When you work with a record in dynamic mode, it is important that you set values in the same order you would within the UI. If you fail to do this, your results may not be accurate. <p>This value is set when the record is created or accessed.</p>
Type	<code>boolean true false (read-only)</code>
Module	N/record Module
Since	Version 2015 Release 2

Syntax



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

Property Description	The record type. This value is set with the <code>record.Type</code> enum during record creation.
Type	string (read-only)
Module	N/record Module
Since	Version 2015 Release 2

Syntax

 **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 recordtype = record.type;
...
//Add additional code
```

record.Sublist

Object Description	Encapsulates a sublist on a standard or custom record. For a complete list of this object's methods and properties, see Sublist Object Members .
Supported Script Types	Client and server-side scripts
Module	N/record Module
Since	Version 2015 Release 2

Syntax

 **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
```

Sublist.getColumn(options)

Method Description	Returns a column in the sublist.
---------------------------	----------------------------------

Returns	record.Column
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 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.	Version 2015 Release 2

Syntax

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

Sublist.id

Property Description	Returns the internal ID of the sublist.
Type	string (read-only)
Module	N/record Module
Since	Version 2015 Release 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 sublistid = objSublist.id;
...
```

```
//Add additional code
```

Sublist.isChanged

Property Description	Indicates whether the sublist has changed on the record form.
Type	boolean true false (read-only)
Module	N/record Module
Since	Version 2015 Release 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
...
if (objSublist.isChanged) {
    ...
}
...
//Add additional code
```

Sublist.isDisplay

Property Description	Indicates whether the sublist is displayed on the record form.
Type	boolean true false (read-only)
Module	N/record Module
Since	Version 2015 Release 2

Syntax

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 (objSublist.isDisplay) {
    ...
}
...
//Add additional code
```

Sublist.type

Property Description	Returns the sublist type.
-----------------------------	---------------------------

Type	string (read-only)
Module	N/record Module
Since	Version 2015 Release 2

Syntax

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 sublisttype = objSublist.type;
...
//Add additional code
```

record.attach(options)

Method Description	Attaches a record to another record.
	<p>Note: For the promise version of this method, see <code>record.attach.promise(options)</code>. Note that promises are only supported in client scripts.</p>
Returns	void
Supported Script Types	Client and server-side scripts
Governance	10 units
Module	N/record Module
Since	Version 2015 Release 2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.record	record.Record	required	The record to attach.	Version 2015 Release 2
options.record.type	string	required	The type of record to attach. Set the value using the <code>record.Type</code> enum. To attach a file from the file	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			cabinet to a record, set type to file.	
options.record.id	number string	required	The internal ID of the record to attach.	Version 2015 Release 2
options.to	record.Record	required	The record that the options.record gets attached to.	Version 2015 Release 2
options.to.type	string	required	The record type of the record to attach to. Set the value using the record.Type enum. To attach a file from the file cabinet to a record, set type to file.	Version 2015 Release 2
options.to.id	number string	required	The internal ID of the record to attach to.	Version 2015 Release 2
options.attributes	Object	optional	The name-value pairs containing attributes for the attachment. By default, this value is null.	Version 2015 Release 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 id = record.attach({
    record: {
        type: 'file',
        id: '200'
    },
    to: {
        type: 'customer',
        id: '90'
    }
});
```

```

    }
});

...
//Add additional code

```

record.attach.promise(options)

Method Description	Attaches a record asynchronously to another record.
	<p>Note: For information about the parameters and errors thrown for this method, see record.attach(options). For more information about promises, see Promise object.</p>
Returns	Promise
Supported Script Types	Client-side scripts
Governance	10 units
Module	N/record Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.record	record.Record	required	The record to attach.	Version 2015 Release 2
options.record.type	string	required	The type of record to attach. Set the value using the record.Type enum. To attach a file from the file cabinet to a record, set type to <code>file</code> .	Version 2015 Release 2
options.record.id	number string	required	The internal ID of the record to attach.	Version 2015 Release 2
options.to	record.Record	required	The record that the options.record gets attached to.	Version 2015 Release 2
options.to.type	string	required	The record type of the record to attach to.	Version 2015 Release 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.	Version 2015 Release 2
<code>options.attributes</code>	Object	optional	The name-value pairs containing attributes for the attachment. By default, this value is null.	Version 2015 Release 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 promise script example, see [Promise object](#).

```
//Add additional code
...
var id = record.attach.promise({
    record: {
        type: 'file',
        id: '200'
    },
    to: {
        type: 'customer',
        id: '90'
    }
});
...
//Add additional code
```

record.copy(options)

Method Description	Creates a new record by copying an existing record in NetSuite.
--------------------	---

	Note: For the promise version of this method, see record.copy.promise(options) . Note that promises are only supported in client scripts.
Returns	<code>record.Record</code>
Supported Script Types	Client and server-side scripts
Governance	Transaction records: 10 usage units Custom records: 2 usage units All other records: 5 usage units
Module	N/record Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.type</code>	string	required	The record type. Set the value using the record.Type enum.	Version 2015 Release 2
<code>options.id</code>	number	required	The internal ID of the existing record instance in NetSuite.	Version 2015 Release 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"> ■ If set to <code>true</code>, the new record is created in dynamic mode. ■ If set to <code>false</code>, the new record is created in standard mode. <p>By default, this value is <code>false</code>.</p> <p>Note: For additional information on standard and dynamic mode, see record.Record.</p>	Version 2015 Release 2
<code>options.defaultValues</code>	Object	optional	Name-value pairs containing default	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			values of fields in the new record. By default, this value is null. For a list of available record default values, see N/record Default Values in the NetSuite Help Center.	

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 objRecord = record.copy({
    type: record.Type.SALES_ORDER,
    id: 157,
    isDynamic: true,
    defaultValues: {
        entity: 107
    }
});
...
//Add additional code
```

record.copy.promise(options)

Method Description	Creates a new record asynchronously by copying an existing record in NetSuite.  Note: For information about the parameters and errors thrown for this method, see record.copy(options) . For more information on promises, see Promise object .
Returns	Promise
Supported Script Types	Client-side scripts
Governance	Transaction records: 10 usage units Custom records: 2 usage units All other records: 5 usage units

Module	N/record Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The record type. Set the value using the record.Type enum .	Version 2015 Release 2
options.id	number	required	The internal ID of the existing record instance in NetSuite.	Version 2015 Release 2
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"> ■ If set to <code>true</code>, the new record is created in dynamic mode. ■ If set to <code>false</code>, the new record is created in standard mode. <p>By default, this value is <code>false</code>.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">  Note: For additional information on standard and dynamic mode, see record.Record. </div>	Version 2015 Release 2
options.defaultValues	Object	optional	Name-value pairs containing default values of fields in the new record. By default, this value is null.	Version 2015 Release 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 promise script example, see [Promise object](#).

```
//Add additional code
...
var objRecord = record.copy.promise({
    type: record.Type.SALES_ORDER,
    id: 157,
    isDynamic: true,
    defaultValues: {
        entity: 107
    }
});
...
//Add additional code
```

record.create(options)

Method Description	Creates a new record. Note: For the promise version of this method, see record.create.promise(options) . Note that promises are only supported in client scripts.
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	Transaction records: 10 usage units Custom records: 2 usage units All other records: 5 usage units
Module	N/record Module
Since	Version 2015 Release 2

Parameters



Note: The options parameter is a JavaScript Object.

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The record type. This value sets the Record.type property of this record. This property is read-only and cannot be changed after the record is created.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			Set the value using the record.Type enum.	
options.isDynamic	boolean true false	optional	<p>Determines whether the new record is created in dynamic mode.</p> <ul style="list-style-type: none"> ■ If set to <code>true</code>, the new record is created in dynamic mode. ■ If set to <code>false</code>, the new record is created in standard mode. <p>By default, this value is <code>false</code>.</p> <p>Note: For additional information on standard and dynamic mode, see record.Record.</p>	Version 2015 Release 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 N/record Default Values in the NetSuite Help Center.</p>	Version 2015 Release 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 objRecord = record.create({
    type: record.Type.SALES_ORDER,
    isDynamic: true,
    defaultValues: {
        entity: 87
    }
});
...
//Add additional code

```

record.create.promise(options)

Method Description	Creates a new record asynchronously.
	<p>Note: For information about the parameters and errors thrown for this method, see record.create(options). For more information on promises, see Promise object.</p>
Returns	Promise
Supported Script Types	Client-side scripts
Governance	Transaction records: 10 usage units Custom records: 2 usage units All other records: 5 usage units
Module	N/record Module
Since	Version 2015 Release 2

Parameters

Note: The options parameter is a JavaScript Object.
--

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The record type. This value sets the Record.type property of this record. This property is read-only and cannot be changed after the record is created. Set the value using the record.Type enum.	Version 2015 Release 2
options.isDynamic	boolean true false	optional	Determines whether the new record is	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<p>created in dynamic mode.</p> <ul style="list-style-type: none"> ■ If set to <code>true</code>, the new record is created in dynamic mode. ■ If set to <code>false</code>, the new record is created in standard mode. <p>By default, this value is <code>false</code>.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Note: For additional information on standard and dynamic mode, see <code>record.Record</code>. </div>	
<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>	Version 2015 Release 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 promise script example, see [Promise object](#).

```
//Add additional code
...
var objRecord = record.create.promise({
    type: record.Type.SALES_ORDER,
    isDynamic: true,
    defaultValues: {
        entity: 87
    }
});
...
//Add additional code
```

record.delete(options)

Method Description	Deletes a record.
	<p>Note: For the promise version of this method, see record.delete.promise(options). Note that promises are only supported in client scripts.</p>
Returns	The internal ID of the deleted record.Record .
Supported Script Types	Client and server-side scripts
Governance	Transaction records: 20 usage units Custom records: 4 usage units All other records: 10 usage units
Module	N/record Module
Since	Version 2015 Release 2

Parameters

i Note: 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> <th>Since</th> </tr> </thead> <tbody> <tr> <td>options.type</td> <td>string</td> <td>required</td> <td>The record type. Set the value using the record.Type enum.</td> <td>Version 2015 Release 2</td> </tr> <tr> <td>options.id</td> <td>number string</td> <td>required</td> <td>The internal ID of the record instance to be deleted.</td> <td>Version 2015 Release 2</td> </tr> </tbody> </table>	Parameter	Type	Required / Optional	Description	Since	options.type	string	required	The record type. Set the value using the record.Type enum.	Version 2015 Release 2	options.id	number string	required	The internal ID of the record instance to be deleted.	Version 2015 Release 2
Parameter	Type	Required / Optional	Description	Since											
options.type	string	required	The record type. Set the value using the record.Type enum.	Version 2015 Release 2											
options.id	number string	required	The internal ID of the record instance to be deleted.	Version 2015 Release 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 objRecord = record.delete({
    type: record.Type.SALES_ORDER,
    id: 128,
});
...
//Add additional code
```

record.delete.promise(options)

Method Description	Deletes a record asynchronously.
	<p>Note: For information about the parameters and errors thrown for this method, see record.delete(options). For more information on promises, see Promise object.</p>
Returns	Promise
Supported Script Types	Client-side scripts
Governance	Transaction records: 20 usage units Custom records: 4 usage units All other records: 10 usage units
Module	N/record Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The record type. Set this value using the record.Type enum .	Version 2015 Release 2
options.id	number string	required	The internal ID of the record instance to be deleted.	Version 2015 Release 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 promise script example, see Promise object .
--

```
//Add additional code
...
var objRecord = record.delete.promise({
    type: record.Type.SALES_ORDER,
    id: 128,
});
...
```

```
//Add additional code
```

record.detach(options)

Method Description	Detaches a record from another record.
	<p>Note: For the promise version of this method, see record.detach.promise(options). Note that promises are only supported in client scripts.</p>
Returns	void
Supported Script Types	Client and server-side scripts
Governance	10 units
Module	N/record Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.record	record.Record	required	The record to be detached.	Version 2015 Release 2
options.record.type	string	required	The type of record to be detached. Set this value using the record.Type enum.	Version 2015 Release 2
options.record.id	number string	required	The ID of the record to be detached.	Version 2015 Release 2
options.from	record.Record	required	The destination record that options.record should be detached from.	Version 2015 Release 2
options.from.type	string	required	The type of the destination. Set this value using the record.Type enum.	Version 2015 Release 2
options.from.id	number string	required	The ID of the destination.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.attributes	Object	optional	Name-value pairs containing default values of fields in the new record. By default, this value is null.	Version 2015 Release 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
...
record.detach({
    record: {
        type: 'file',
        id:'200'
    },
    from: {
        type: 'customer',
        id:'90'
    }
})
...
//Add additional code
```

record.detach.promise(options)

Method Description	Detaches a record from another record asynchronously.
	 Note: For information about the parameters and errors thrown for this method, see record.detach(options) . For more information on promises, see Promise object .
Returns	Promise
Supported Script Types	Client-side scripts
Governance	10 units
Module	N/record Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.record	record.Record	required	The record to be detached.	Version 2015 Release 2
options.record.type	string	required	The type of record to be detached. Set this value using the record.Type enum.	Version 2015 Release 2
options.record.id	number string	required	The ID of the record to be detached.	Version 2015 Release 2
options.from	record.Record	required	The destination record that options.record should be detached from.	Version 2015 Release 2
options.from.type	string	required	The type of the destination. Set this value using the record.Type enum.	Version 2015 Release 2
options.from.id	number string	required	The ID of the destination.	Version 2015 Release 2
options.attributes	Object	optional	Name-value pairs containing default values of fields in the new record. By default, this value is null.	Version 2015 Release 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 promise script example, see Promise object .
--

```
//Add additional code
```

```

...
record.detach.promise({
    record: {
        type: 'file',
        id:'200'
    },
    from: {
        type: 'customer',
        id:'90'
    }
})
...
//Add additional code

```

record.load(options)

Method Description	Loads an existing record.
	Note: For the promise version of this method, see record.load.promise(options) . Note that promises are only supported in client scripts.
Returns	<code>record.Record</code>
Supported Script Types	Client and server-side scripts
Governance	Transaction records: 10 usage units Custom records: 2 usage units All other records: 5 usage units
Module	N/record Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.type</code>	string	required	The type of record to load. This value sets the Record.type property for the record. Set this value using the record.Type enum.	Version 2015 Release 2
<code>options.id</code>	number	required	The internal ID of the existing record instance in NetSuite. The internal ID of the record is displayed on	

Parameter	Type	Required / Optional	Description	Since
			the list page for the record type.	
options.isDynamic	boolean true false	optional	<p>Determines whether the record is loaded in dynamic mode.</p> <ul style="list-style-type: none"> ■ If set to <code>true</code>, the record is loaded in dynamic mode. ■ If set to <code>false</code>, the record is loaded in standard mode. <p>By default, this value is <code>false</code>.</p> <p>Note: For additional information on standard and dynamic mode, see <code>record.Record</code>.</p>	Version 2015 Release 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 N/record Default Values 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
...
var objRecord = record.load({
    type: record.Type.SALES_ORDER,
    id: 157,
```

```

    isDynamic: true,
});
...
//Add additional code

```

record.load.promise(options)

Method Description	Loads an existing record asynchronously.
	<p>Note: For information about the parameters and errors thrown for this method, see record.load(options). For more information on promises, see Promise object.</p>
Returns	Promise
Supported Script Types	Client-side scripts
Governance	Transaction records: 10 usage units Custom records: 2 usage units All other records: 5 usage units
Module	N/record Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The type of record to load. This value sets the Record.type property for the record. Set this value using the record.Type enum.	Version 2015 Release 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 true false	optional	Determines whether the record is loaded in dynamic mode. <ul style="list-style-type: none"> ■ If set to <code>true</code>, the record is loaded in dynamic mode. 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> ■ If set to <code>false</code>, the record is loaded in standard mode. By default, this value is <code>false</code>. <div style="border: 1px solid #ccc; padding: 10px; background-color: #f0f8ff; margin-top: 10px;"> Note: For additional information on standard and dynamic mode, see <code>record.Record</code>. </div>	
<code>options.defaultValues</code>	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
<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 promise script example, see [Promise object](#).

```
//Add additional code
...
var objRecord = record.load.promise({
    type: record.Type.SALES_ORDER,
    id: 157,
    isDynamic: true,
});
...
//Add additional code
```

record.submitFields(options)

Method Description	Updates and submits one or more body fields on an existing record in NetSuite, and returns the internal ID of the parent record. When you use this method, you do not need to load or submit the parent record. You can use this method to edit and submit the following:
---------------------------	--

	<ul style="list-style-type: none"> ■ Standard body fields that support inline editing (direct list editing). For more information, see the help topic Using Inline Editing. ■ Custom body fields that support inline editing. <p>You cannot use this method to edit and submit the following:</p> <ul style="list-style-type: none"> ■ Select fields ■ Sublist line item fields ■ Subrecord fields (for example, address fields) <div style="background-color: #e0f2ff; border-radius: 10px; padding: 10px; margin-top: 10px;"> Note: For the promise version of this method, see <code>record.submitFields.promise(options)</code>. Note that promises are only supported in client scripts. </div>
Returns	The internal ID of the parent record.
Supported Script Types	Client and server-side scripts
Governance	Transaction records: 10 usage units Custom records: 2 usage units All other records: 5 usage units
Module	N/record Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The type of record. This value sets the Record.type property for the record. Set this value using the record.Type enum.	Version 2015 Release 2
options.id	number string	required	The internal ID of the existing record instance in NetSuite.	Version 2015 Release 2
options.values	Object	required	The ID-value pairs for each field you want to edit and submit.	Version 2015 Release 2
options.options	Object	optional	Additional options to set for the record.	Version 2015 Release 2
options.options.enablesourcing	boolean true false	optional	Indicates whether to enable sourcing during the record update. By default, this value is true.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.options.ignoreMandatoryFields	boolean true false	optional	Indicates whether to ignore mandatory fields during record submission. By default, this value is false.	Version 2015 Release 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 id = record.submitFields({
    type: record.Type.SALES_ORDER,
    id: 6,
    values: {
        memo: 'ABC'
    },
    options: {
        enableSourcing: false,
        ignoreMandatoryFields : true
    }
});
...
//Add additional code
```

record.submitFields.promise(options)

Method Description	<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. When you use this method, you do not need to load or submit the parent record.</p> <p>You can use this method to edit and submit the following:</p> <ul style="list-style-type: none"> ■ Standard body fields that support inline editing (direct list editing). For more information, see the help topic Using Inline Editing. ■ Custom body fields that support inline editing. <p>You cannot use this method to edit and submit the following:</p> <ul style="list-style-type: none"> ■ Select fields
---------------------------	--

	<ul style="list-style-type: none"> ■ Sublist line item fields ■ Subrecord fields (for example, address fields) <p>Note: For information about the parameters and errors thrown for this method, see <code>record.submitFields(options)</code>. For more information on promises, see Promise object.</p>
Returns	Promise
Supported Script Types	Client-side scripts
Governance	Transaction records: 10 usage units Custom records: 2 usage units All other records: 5 usage units
Module	N/record Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The type of record. This value sets the <code>Record.type</code> property for the record. Set this value using the <code>record.Type</code> enum.	Version 2015 Release 2
options.id	number string	required	The internal ID of the existing record instance in NetSuite.	Version 2015 Release 2
options.values	Object	required	The ID-value pairs for each field you want to edit and submit.	Version 2015 Release 2
options.options	Object	optional	Additional options to set for the record.	Version 2015 Release 2
options.options.enablesourcing	boolean true false	optional	Indicates whether to enable sourcing during the record update. By default, this value is <code>true</code> .	Version 2015 Release 2
options.options.ignoreMandatoryFields	boolean true false	optional	Indicates whether to ignore mandatory fields during record submission. By default, this value is <code>false</code> .	Version 2015 Release 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 promise script example, see [Promise object](#).

```
//Add additional code
...
var id = record.submitFields.promise({
    type: record.Type.SALES_ORDER,
    id: 6,
    values: {
        memo: 'ABC'
    },
    options: {
        enableSourcing: false,
        ignoreMandatoryFields: true
    }
});
...
//Add additional code
```

record.transform(options)

Method Description	Transforms a record from one type into another, using data from an existing record. You can use this method to automate order processing, creating item fulfillment transactions and invoices off of orders. For a list of supported transformations, see Supported Transformation Types .
Returns	record.Record
Supported Script Types	Client and server-side scripts
Governance	Transaction records: 10 usage units Custom records: 2 usage units All other record types: 5 usage units
Module	N/record Module
Since	Version 2015 Release 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 Record.type property for the record. This property is read-only and cannot be changed after the record is loaded. Set this value using the record.Type .	Version 2015 Release 2
options.fromId	number	required	The internal ID of the existing record instance being transformed.	Version 2015 Release 2
options.toType	string	required	The record type of the record returned when the transformation is complete.	Version 2015 Release 2
options.isDynamic	boolean true false	optional	<p>Determines whether the new record is created in dynamic mode.</p> <ul style="list-style-type: none"> ■ If set to <code>true</code>, the new record is created in dynamic mode. ■ If set to <code>false</code>, the new record is created in standard mode. <p>By default, this value is <code>false</code>.</p> <div style="background-color: #e0f2ff; padding: 5px; border-radius: 5px; margin-top: 10px;"> Note: For additional information on standard and dynamic mode, see record.Record. </div>	Version 2015 Release 2
options.defaultValue	Object	optional	Name-value pairs containing default	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			values of fields in the new record. By default, this value is null. For a list of available record default values, see N/record Default Values in the NetSuite Help Center.	

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 objRecord = record.transform({
    fromType: record.Type.CUSTOMER,
    fromId: 107,
    toType: record.Type.SALES_ORDER,
    isDynamic: true,
});
...
//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
Customer	Quote
Customer	Invoice
Customer	Opportunity
Customer	Sales Order
Employee	Expense Report

Original Record Type	Transformed Record Type
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
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
<p>Note: The return authorization must be approved and received for this transform to work.</p>	
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

Original Record Type	Transformed Record Type
Vendor Bill	Vendor Return Authorization
Vendor Return Authorization	Item Fulfillment
Vendor Return Authorization	Vendor Credit
Work Order	Assembly Build

record.transform.promise(options)

Method Description	<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> <p>For a list of supported transformations, see Supported Transformation Types.</p> <p>Note: For information about the parameters and errors thrown for this method, see record.transform(options). For more information on promises, see Promise object.</p>
Returns	Promise
Supported Script Types	Client-side scripts
Governance	Transaction records: 10 usage units Custom records: 2 usage units All other record types: 5 usage units
Module	N/record Module
Since	Version 2015 Release 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 Record.type property for the record. This property is read-only and cannot be changed after the record is loaded. Set this value using the record.Type .	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.fromId	number	required	The internal ID of the existing record instance being transformed.	Version 2015 Release 2
options.toType	string	required	The record type of the record returned when the transformation is complete.	Version 2015 Release 2
options.isDynamic	boolean true false	optional	<p>Determines whether the new record is created in dynamic mode.</p> <ul style="list-style-type: none"> ■ If set to <code>true</code>, the new record is created in dynamic mode. ■ If set to <code>false</code>, the new record is created in standard mode. <p>By default, this value is <code>false</code>.</p> <div style="border: 1px solid #ccc; padding: 5px; background-color: #f0f8ff; margin-top: 10px;"> Note: For additional information on standard and dynamic mode, see <code>record.Record</code>. </div>	Version 2015 Release 2
options.defaultValue	Object	optional	Name-value pairs containing default values of fields in the new record. By default, this value is null.	Version 2015 Release 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 promise script example, see [Promise object](#).

```
//Add additional code
```

```

...
var objRecord = record.transform.promise({
    fromType: record.Type.CUSTOMER,
    fromId: 107,
    toType: record.Type.SALES_ORDER,
    isDynamic: true,
});
...
//Add additional code

```

record.Type

Enum Description	Enumeration that holds the string values for supported record types. This enum is used to set the value of the <code>Record.type</code> property. Note that the <code>Record.type</code> property is read only. Its value must be set using this enum.
Module	N/record Module
Since	Version 2015 Release 2

Values

- | | | |
|--|--|---|
| <ul style="list-style-type: none"> ■ ACCOUNT ■ ACCOUNTING_BOOK ■ ACCOUNTING_PERIOD ■ AMORTIZATION_SCHEDULE ■ AMORTIZATION_TEMPLATE ■ ASSEMBLY_BUILD ■ ASSEMBLY_ITEM ■ ASSEMBLY_UNBUILD ■ BILLING_ACCOUNT ■ BILLING_CLASS ■ BILLING_SCHEDULE ■ BIN ■ BIN_TRANSFER ■ BIN_WORKSHEET ■ BLANKET_PURCHASE_ORDER ■ BUNDLE_INSTALLATION_SCRIPT ■ CALENDAR_EVENT ■ CAMPAIGN | <ul style="list-style-type: none"> ■ INTER_COMPANY_TRANSFER_ORDER ■ INVENTORY_ADJUSTMENT ■ INVENTORY_COST_REVALUATION ■ INVENTORY_COUNT ■ INVENTORY_DETAIL ■ INVENTORY_ITEM ■ INVENTORY_NUMBER ■ INVENTORY_TRANSFER ■ INVOICE ■ ISSUE ■ ITEM_ACCOUNT_MAPPING ■ ITEM_DEMAND_PLAN ■ ITEM_FULFILLMENT ■ ITEM_GROUP ■ ITEM_RECEIPT ■ ITEM_REVISION ■ ITEM_SUPPLY_PLAN | <ul style="list-style-type: none"> ■ RATE_PLAN ■ REALLOCATE_ITEM ■ RESOURCE_ALLOCATION ■ RESTLET ■ RETURN_AUTHORIZATION ■ REVENUE_ARRANGEMENT ■ REVENUE_COMMITMENT ■ REVENUE_COMMITMENT_REVERSAL ■ REVENUE_PLAN ■ REV_REC_SCHEDULE ■ REV_REC_TEMPLATE ■ SALES_ORDER ■ SALES_TAX_ITEM ■ SCHEDULED_SCRIPT ■ SCHEDULED_SCRIPT_INSTANCE ■ SCRIPT_DEPLOYMENT ■ SERIALIZED_ASSEMBLY_ITEM |
|--|--|---|

■ CAMPAIGN_TEMPLATE	■ JOB	■ SERIALIZED_INVENTORY_ITEM
■ CASH_REFUND	■ JOB_REQUSITION	■ SERVICE_ITEM
■ CASH_SALE	■ JOURNAL_ENTRY	■ SHIP_ITEM
■ CHARGE	■ KIT_ITEM	■ SOLUTION
■ CHECK	■ KUDOS	■ STATISTICAL_JOURNAL_ENTRY
■ CLASSIFICATION	■ LEAD	■ SUBSCRIPTION
■ CLIENT_SCRIPT	■ LOCATION	■ SUBSCRIPTION_CHANGE_ORDER
■ COMPETITOR	■ LOT_NUMBERED_ASSEMBLY_ITEM	■ SUBSCRIPTION_LINE
■ CONTACT	■ LOT_NUMBERED_INVENTORY_ITEM	■ SUBSCRIPTION_PLAN
■ COUPON_CODE	■ MANUFACTURING_COST_TEMPLATE	■ SUBSIDIARY
■ CREDIT_CARD_CHARGE	■ MANUFACTURING_OPERATION_TASK	■ SUBTOTAL_ITEM
■ CREDIT_CARD_REFUND	■ MANUFACTURING_ROUTING	■ SUITELET
■ CREDIT_MEMO	■ MAP_REDUCE_SCRIPT	■ SUPPORT_CASE
■ CURRENCY	■ MARKUP_ITEM	■ TASK
■ CUSTOMER	■ MASSUPDATE_SCRIPT	■ TAX_ACCT
■ CUSTOMER_CATEGORY	■ MESSAGE	■ TAX_GROUP
■ CUSTOMER_DEPOSIT	■ MFG_PLANNED_TIME	■ TAX_PERIOD
■ CUSTOMER_PAYMENT	■ NEXUS	■ TAX_TYPE
■ CUSTOMER_REFUND	■ NON_INVENTORY_ITEM	■ TERM
■ CUSTOM_TRANSACTION	■ NOTE	■ TERMINATION_REASON
■ DEPARTMENT	■ OPPORTUNITY	■ TIME_BILL
■ DEPOSIT	■ ORDER_SCHEDULE	■ TIME_OFF_CHANGE
■ DEPOSIT_APPLICATION	■ ORGANIZATION_VALUE	■ TIME_OFF_PLAN
■ DESCRIPTION_ITEM	■ OTHER_CHARGE_ITEM	■ TIME_OFF_REQUEST
■ DISCOUNT_ITEM	■ OTHER_GOVERNMENT_ISSUED_ID	■ TIME_OFF_RULE
■ DOWNLOAD_ITEM	■ OTHER_NAME	■ TIME_OFF_TYPE
■ DRIVERS_LICENSE	■ PARTNER	■ TOPIC
■ EMAIL_TEMPLATE	■ PASSPORT	■ TRANSFER_ORDER
■ EMPLOYEE	■ PAYCHECK_JOURNAL	■ UNITS_TYPE
■ ENTITY_ACCOUNT_MAPPING	■ PAYMENT_ITEM	■ USEREVENT_SCRIPT
■ ESTIMATE	■ PAYROLL_ITEM	■ VENDOR
■ EXPENSE_CATEGORY	■ PHONE_CALL	■ VENDOR_BILL
■ EXPENSE_REPORT	■ PORTLET	■ VENDOR_CATEGORY
■ FAIR_VALUE_PRICE	■ POSITION	■ VENDOR_CREDIT
■ FOLDER	■ PRICE_LEVEL	■ VENDOR_PAYMENT
■ GENERIC_RESOURCE	■ PROJECT_EXPENSE_TYPE	■ VENDOR_RETURN_AUTHORIZATION
■ GIFT_CERTIFICATE	■ PROJECT_TASK	■ WEBSITE
■ GIFT_CERTIFICATE_ITEM	■ PROJECT_TEMPLATE	■ WORKFLOW_ACTION_SCRIPT
■ GLOBAL_ACCOUNT_MAPPING		
■ GOVERNMENT_ISSUED_ID_TYPE		

- | | | |
|--|--|---|
| <ul style="list-style-type: none"> ■ HCM_JOB ■ INTER_COMPANY_JOURNAL_ENTRY | <ul style="list-style-type: none"> ■ PROMOTION_CODE ■ PROSPECT ■ PURCHASE_CONTRACT ■ PURCHASE_ORDER ■ PURCHASE_REQUSITION | <ul style="list-style-type: none"> ■ WORK_ORDER ■ WORK_ORDER_CLOSE ■ WORK_ORDER_COMPLETION ■ WORK_ORDER_ISSUE |
|--|--|---|

Syntax

⚠ 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

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

i Note: Suitelets, beforeLoad user events, and synchronous afterSubmit user events are supported. Some module members support client-side scripts (See individual member topics listed below.). This module does not support beforeSubmit and asynchronous afterSubmit user events.

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

N/redirect Module Members

Member Type	Name	Return Type	Description
Method	redirect.redirect(options)	void	Redirects to the URL of a Suitelet that is available externally (available without login).
	redirect.toRecord(options)	void	Redirects to a NetSuite record.

Member Type	Name	Return Type	Description
	redirect.toSavedSearch(options)	void	Redirects to a saved search.
	redirect.toSavedSearchResult(options)	void	Redirects to a saved search result.
	redirect.toSearch(options)	void	Redirects to search.
	redirect.toSearchResult(options)	void	Redirects to search results.
	redirect.toSuitelet(options)	void	Redirects to a Suitelet.
	redirect.toTaskLink(options)	void	Redirects to a tasklink.

N/redirect Module Script Sample

The following example sets the redirect URL to a newly created task record. To set the redirect using the record id, the record must have been previously submitted.

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

```
/** 
 * @NApiVersion 2.x
 */
require(['N/record', 'N/redirect'],
    function(record, redirect) {
        function redirectToTaskRecord() {
            var taskTitle = 'New Opportunity';
            var taskRecord = record.create({
                type: record.Type.TASK
            });
            taskRecord.setValue('title', taskTitle);
            var taskRecordId = taskRecord.save();
            redirect.toRecord({
                type: record.Type.TASK,
                id: taskRecordId
            });
            redirectToTaskRecord();
        });
    });
});
```

redirect.redirect(options)

Method Description	Method used to set the redirect to the URL of a Suitelet that is available externally (Suitelets set to Available Without Login on the Script Deployment page).
Returns	Void

Supported Script Types	Suitelets, beforeLoad user events, and synchronous afterSubmit user events
Governance	None
Module	N/redirect Module
Since	Version 2015 Release 2

Parameters

<p>Note: The options parameter is a JavaScript object.</p>			
Parameter	Type	Required / Optional	Description
options.url	string	required	<p>The URL of a Suitelet that is available externally</p> <p>Note: For an external URL, Available without Login must be enabled on the Script Deployment page for the Suitelet.</p>
options.parameters	Object	optional	Contains additional URL parameters as key/value pairs.

Syntax

<p>Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/redirect Module Script Sample.</p> <pre>//Add additional code ... redirect.redirect({ url: '/app/site/hosting/scriptlet.nl?script=130&deploy=1', parameters: {'custparam_test':'helloWorld'} }); ... //Add additional code</pre>

redirect.toRecord(options)

Method Description	Method used to set the redirect URL to a specific NetSuite record.
	<p>Note: If you redirect a user to a record, the record must first exist in NetSuite. If you want to redirect a user to a new record, you must first create and submit the record before redirecting them. You must also ensure that any required fields for the new record are populated before submitting the record.</p>
Returns	Void

Supported Script Types	Suitelets, beforeLoad user events, and synchronous afterSubmit user events
Governance	None
Module	N/redirect Module
Since	Version 2015 Release 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.
options.isEditMode	boolean true false	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.
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 [N/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)

Method Description	Method used to load an existing saved search and redirect to the populated search definition page.
---------------------------	--

Returns	Void
Supported Script Types	Client scripts and afterSubmit user event scripts
Governance	5 units
Module	N/redirect Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
options.id	number	required	<p>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.</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/redirect Module Script Sample](#).

```
//Add additional code
...
redirect.toSavedSearch({id: 234});
...
//Add additional code
```

redirect.toSavedSearchResult(options)

Method Description	Method used to redirect a user to a search results page for an existing saved search.
Returns	Void
Supported Script Types	Client scripts and afterSubmit user event scripts
Governance	5 units
Module	N/redirect Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
options.id	number	required	<p>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.</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/redirect Module Script Sample](#).

```
//Add additional code
...
redirect.toSavedSearchResult({id: 234});
...
//Add additional code
```

redirect.toSearch(options)

Method Description	Method used to redirect a user to an ad-hoc search built in SuiteScript. This method loads a search into the session, and then redirects to a URL that loads the search definition page.
Returns	Void
Supported Script Types	Client scripts and afterSubmit user event scripts
Governance	None
Module	N/redirect Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
options.search	<code>search.Search</code>	required	

Syntax



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

redirect.toSearchResult(options)

Method Description	Method used to redirect a user to a search results page. For example, the results from an ad-hoc search created with the N/search Module , or a loaded search that you modified but did not save.
Returns	Void
Supported Script Types	Client scripts and afterSubmit user event scripts
Governance	None
Module	N/redirect Module
Since	Version 2015 Release 2

Parameters



Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.Search	search.Search	required	

Syntax



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

redirect.toSuitelet(options)

Method Description	Method used to redirect the user to a Suitelet.
Returns	Void
Supported Script Types	Suitelets, beforeLoad user events, and synchronous afterSubmit user events
Governance	None
Module	N/redirect Module
Since	Version 2015 Release 2

Parameters

i Note: The options parameter is a JavaScript object.

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 true false	optional	The default value is false – indicates an external Suitelet URL.
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 [N/redirect Module Script Sample](#).

```
//Add additional code
...
redirect.toSuitelet({
 scriptId: 31,
  deploymentId: 1,
  parameters: {'custparam_test':'helloWorld'}
});
...
//Add additional code
```

redirect.toTaskLink(options)

Method Description	Method used to redirect a user to a tasklink.
Returns	Void
Supported Script Types	Suitelets, beforeLoad user events, and synchronous afterSubmit user events
Governance	None
Module	N/redirect Module
Since	Version 2015 Release 2

Parameters

i Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.id	string	required	The taskId for a tasklink

Parameter	Type	Required / Optional	Description
			For a list of supported task IDs, see the help topic Task IDs .
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 [N/redirect Module Script Sample](#).

```
//Add additional code
...
redirect.toTaskLink({
    id: ADMI_SHIPPING ,
    parameters: {'custparam_test':'helloWorld'}
});
...
//Add additional code
```

N/render Module

The render module encapsulates functionality for printing, PDF creation, form creation from templates, and email creation from templates.

- [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	Description
Object	render.EmailMergeResult	Object	Encapsulates an email merge result.
	render.TemplateRenderer	Object	Encapsulates a template engine object that produces HTML and PDF printed forms utilizing advanced PDF/HTML template capabilities.
Method	render.bom(options)	file.File	Creates a PDF or HTML file object containing a bill of materials.

Member Type	Name	Return Type / Value Type	Description
	render.create()	render.TemplateRenderer	Creates a render.TemplateRenderer object
	render.mergeEmail(options)	render.EmailMergeResult	Creates a render.EmailMergeResult object
	render.packingSlip(options)	file.File	Creates a PDF or HTML file object containing a packing slip.
	render.pickingTicket(options)	file.File	Creates a PDF or HTML file object containing a picking ticket.
	render.statement(options)	file.File	Creates a PDF or HTML file object containing a statement.
	render.transaction(options)	file.File	Creates a PDF or HTML file object containing a transaction.
	render.xmlToPdf(options)	file.File	Passes XML to the BFO tag library (which is stored by NetSuite), and returns a PDF file.
Enum	render.DataSource	enum	Holds the string values for supported data source types.
	render.PrintMode	enum	Holds the string values for supported print output types.

EmailMergeResult Object Members

Member Type	Name	Return Type / Value Type	Description
Property	EmailMergeResult.body	string (read-only)	The body of the email distribution in string format
	EmailMergeResult.subject	string (read-only)	The subject of the email distribution in string format

TemplateRenderer Object Members

Member Type	Name	Return Type / Value Type	Description
Method	TemplateRenderer.addCustomDataSource(options)	void	Adds to an advanced template an XML file or JSON object as custom data source

Member Type	Name	Return Type / Value Type	Description
	TemplateRenderer.addRecord(options)	void	Binds a record to a template variable.
	TemplateRenderer.addSearchResults(options)	void	Binds a search result to a template variable.
	TemplateRenderer.renderAsPdf()	Object	Uses an advanced template to produce a PDF printed form
	TemplateRenderer.renderPdfToResponse()	void	Renders PDF template content as a server response.
	TemplateRenderer.renderAsString()	string	Returns template content in string form
	TemplateRenderer.setTemplateById(options)	void	Sets the template using the internal ID
	TemplateRenderer.setTemplateByscriptId(options)	void	Sets the template using the script ID
	TemplateRenderer.renderToResponse(options)	void	Renders HTML template content as a server response.
Property	TemplateRenderer.templateContent	string	Content of template

N/render Module Script Sample

i 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 [SuiteScript 2.0 – Script Architecture](#) and [SuiteScript 2.0 Script Types and Entry Points](#).

The following example generates 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="18">\nHello World!\n</body>\n</pdf>';
            var pdfFile = render.xmlToPdf({
                xmlString: xmlStr
            });
            generatePdfFileFromRawXml();
        });
    });
});
```

The following example renders a transaction record into a 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();  
    });
```

The following example renders an invoice into a pdf file using an xml template in the file cabinet. This example requires the Advanced PDF/HTML Templates feature.

```
/**  
 * @NApiVersion 2.x  
 */  
// This example shows how to render an invoice into a pdf file using an xml template in the file cabinet.  
// Note that this example requires the Advanced PDF/HTML Templates feature  
require(['N/render', 'N/file', 'N/record'],  
    function(render, file, record) {  
        function renderRecordToPdfWithTemplate() {  
            var xmlTemplateFile = file.load('Templates/PDF Templates/invoicePDFTemplate.xml');  
            var renderer = render.create();  
            renderer.templateContent = xmlTemplateFile.getContents();  
            renderer.addRecord('grecord', record.load({  
                type: record.Type.INVOICE,  
                id: 37  
            }));  
            var invoicePdf = renderer.renderAsPdf();  
        }  
        renderRecordToPdfWithTemplate();  
    });
```

In the preceding example, the invoicePDFTemplate.xml file was referenced in the File Cabinet. This file is similar to the Standard Invoice PDF/HTML Template found in Customization > Forms > Advanced PDF/HTML Templates.

The following example renders search results into a pdf file.

Note: This sample script uses the `define` function. The NetSuite Debugger cannot step through a `define` function. If you need to step through your code in the NetSuite Debugger, you must use a `require` function.

```
/**  
 * @NApiVersion 2.0  
 * @NScriptType suitelet
```

```

/*
// This example shows how to render search results into a pdf file.
// Note that this sample is a Suitelet, so it cannot be run in the debugger.
define(['N/render', 'N/search'],
    function(render, search) {
        function onRequest: function(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.t
tf\" " +
                "src-italic=\"NetSuiteFonts/verdanai.ttf\" " + "src-bolditalic=\"NetSuiteFo
nts/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

Object Description	Encapsulates an email merge result. Use render.mergeEmail(options) to create and return this object. For a complete list of this object's properties, see EmailMergeResult Object Members .
Supported Script Types	Server-side scripts
Module	N/render Module
Since	Version 2015 Release 2

Syntax



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

Property Description	The body of the email distribution in string format
Type	string (read-only)
Module	N/render Module
Since	Version 2015 Release 2

Syntax



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

Property Description	The subject of the email distribution in string format
Type	string (read-only)
Module	N/render Module
Since	Version 2015 Release 2

Syntax



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

Object Description	Encapsulates a template engine object that produces HTML and PDF printed forms utilizing advanced PDF/HTML template capabilities. The template engine object includes methods that pass in a template as string to be interpreted by FreeMarker, and render interpreted content in your choice of two different formats: as HTML output to an <code>nlobjResponse</code> object, or as XML string that can be passed to <code>render.xmlToPdf(options)</code> to produce a PDF. This object is available when the Advanced PDF/HTML Templates feature is enabled. For a complete list of this object's methods and properties, see TemplateRenderer Object Members .
Supported Script Types	Server-side scripts
Module	N/render Module
Since	Version 2015 Release 2

Syntax



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 xmlTmpFile = file.load(4575);
var myFile = render.create();
myFile.templateContent = xmlTmpFile.getContents();
myFile.addRecord({
    templateName: 'invoicePDFTemplate.xml',
    record: 1234
});
var customPdf = renderer.renderAsPdf()
...
//Add additional code

```

TemplateRenderer.addCustomDataSource(options)

Method Description	Adds XML or JSON as custom data source to an advanced PDF/HTML template
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/render Module
Since	Version 2016 Release 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.addRecord(options)

Method Description	Binds a record to a template variable.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/render Module
Since	Version 2015 Release 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)

Method Description	Binds a search result to a template variable.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/render Module
Since	Version 2015 Release 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](#).

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()

Method Description	Uses the advanced template to produce a PDF printed form
Returns	file.File
Supported Script Types	Server-side scripts
Governance	None
Module	N/render Module
Since	Version 2015 Release 2

Syntax



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()

Method Description	Renders a server response into a PDF file. For example, you can pass in a response to be rendered as a PDF in a browser, or downloaded by a user.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/render Module

Since	Version 2015 Release 2
-------	------------------------

Parameters

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

Parameter	Type	Required / Optional	Description
response	http.ServerResponse	required	Response that will be written to PDF. For example, the response passed from a Suitelet.

Syntax

 **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.renderPdfToResponse({
    response: myServerResponseObj
});
...
//Add additional code
```

TemplateRenderer.renderAsString()

Method Description	Return template content in string form
Returns	string
Supported Script Types	Server-side scripts
Governance	None
Module	N/render Module
Since	Version 2015 Release 2

Syntax

 **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.renderAsString();
```

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

TemplateRenderer.renderToResponse(options)

Method Description	Writes template content to a server response.
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/render Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
options.response	http.ServerResponse	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)

Method Description	Sets the template using the internal ID
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/render Module
Since	Version 2016 Release 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
```

TemplateRenderer.setTemplateByscriptId(options)

Method Description	Sets the template using the script ID
Returns	Void
Supported Script Types	Server-side scripts
Governance	None
Module	N/render Module
Since	Version 2016 Release 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("STDTMPLPRICELIST");
var xml = renderer.renderAsString();
...
//Add additional code

```

TemplateRenderer.templateContent

Property Description	Content of template
Type	string
Module	N/render Module
Since	Version 2015 Release 2

Syntax



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)

Method Description	Use this method to create a PDF or HTML object of a bill of material.
Returns	file.File that contains a PDF or HTML document
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/render Module
Since	Version 2015 Release 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

Parameter	Type	Required / Optional	Description
options.printMode	string	optional	The print output type. Set using the render.PrintMode enum. By default, uses the company/user preference for print output.

Syntax

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

render.create()

Method Description	Use this method to produce HTML and PDF printed forms with advanced PDF/HTML templates. Creates render.TemplateRenderer . This object includes methods that pass in a template as string to be interpreted by FreeMarker, and render interpreted content in your choice of two different formats: as HTML output to http.ServerResponse , or as XML string that can be passed to render.xmlToPdf(options) to produce a PDF.
Returns	render.TemplateRenderer
Supported Script Types	Server-side scripts
Governance	None
Module	N/render Module
Since	Version 2015 Release 2

Syntax

 **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)

Method Description	Creates a <code>render.EmailMergeResult</code> object for a mail merge with an existing scriptable email template
Returns	<code>render.EmailMergeResult</code>
Supported Script Types	Server-side scripts
Governance	None
Module	N/render Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.templateId</code>	number	required	Internal ID of the template
<code>options.entity</code>	RecordRef	required	Entity
<code>options.recipient</code>	RecordRef	required	Recipient
<code>options.customRecord</code>	RecordRef	required	Custom record
<code>options.supportCaseId</code>	number	required	Support case ID
<code>options.transactionId</code>	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
<code>RecordRef.id</code>	number	required	Internal ID of the record instance
<code>RecordRef.type</code>	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)

Method Description	Use this method to create a PDF or HTML object of a packing slip.
Returns	<code>file.File</code> that contains a PDF or HTML document
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/render Module
Since	Version 2015 Release 2

Parameters

i Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
<code>options.entityId</code>	number	required	The internal ID of the packing slip to print
<code>options.printMode</code>	string	optional	The print output type. Set using the <code>render.PrintMode</code> enum.

Parameter	Type	Required / Optional	Description
			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

Syntax



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

render.pickingTicket(options)

Method Description	Use this method to create a PDF or HTML object of a picking ticket.
Returns	file.File that contains a PDF or HTML document
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/render Module
Since	Version 2015 Release 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 render.PrintMode 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

Parameter	Type	Required / Optional	Description
options.location	number	optional	Location for the ticket

Syntax

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

render.statement(options)

Method Description	Use this method to create a PDF or HTML object of a statement.
Returns	file.File that contains a PDF or HTML document
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/render Module
Since	Version 2015 Release 2

Parameters

 **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 render.PrintMode 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
options.startDate	Date	optional	Date of the oldest transaction to appear on the statement
options.statementDate	Date	optional	Statement date

Parameter	Type	Required / Optional	Description
options.openTransactionsOnly	boolean true false	optional	Include only open transactions

Syntax

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.createStatement({
    entityId: 23,
    printMode: render.PrintMode.HTML
});
...
//Add additional code
```

render.transaction(options)

Method Description	Use this method to create a PDF or HTML object of a transaction.
	<p>Note: File size is limited to 10MB.</p> <p>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 Advanced PDF/HTML Templates</p>
Returns	file.File that contains a PDF or HTML document
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/render Module
Since	Version 2015 Release 2

Parameters

Note: 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 render.PrintMode enum.

Parameter	Type	Required / Optional	Description
			By default, uses the company/user preference for print output.
options.formId	number	optional	The transaction form number

Syntax

 **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.transaction({
    entityId: 23,
    printMode: render.PrintMode.HTML
});
...
//Add additional code
```

render.xmlToPdf(options)

Method Description	Method used to pass XML to the Big Faceless Organization tag library (which is stored by NetSuite), and return a PDF file.
	 Note: File size cannot exceed 10MB.
Returns	file.File
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/render Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
options.xmlString	xml.Document string	required	XML document or string to convert to 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
```

```

...
var pdfFile = render.xmlToPdf({
    xmlString: xmlStr
});
...
//Add additional code

```

render.DataSource

Enum Description	Holds the string values for supported data source types. Use this enum to set the <code>options.format</code> parameter.
Module	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

Enum Description	Holds the string values for supported print output types. Use this enum to set the <code>options.printMode</code> parameter.
-------------------------	--

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

Load the runtime module when you want to access the current runtime settings for the script and script deployment, the user currently executing the script, and user-defined sessions.

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

- N/runtime Module Members
- Script Object Members
- Session Object Members
- User Object Members
- [N/runtime Module Script Sample](#)

N/runtime Module Members

Member Type	Name	Return Type / Value Type	Description
Object	runtime.Script	Object	Encapsulates the runtime settings of the currently executing script.
	runtime.Session	Object	Encapsulates the user session for the currently executing script.

Member Type	Name	Return Type / Value Type	Description
	runtime.User	Object	Encapsulates the properties and preferences for the user of the currently executing script.
Method	runtime.getCurrentScript()	runtime.Script	Returns a runtime.Script that represents the currently executing script.
	runtime.getCurrentSession()	runtime.Session	Returns a runtime.Session that represents the user session for the currently executing script.
	runtime.getCurrentUser()	runtime.User	Returns a runtime.User that represents the properties and preferences for the user of the currently executing script.
	runtime.isFeatureInEffect(options)	boolean true false	Use this method to determine if a particular feature is enabled in a NetSuite account.
Property	runtime.accountId	string (read-only)	Returns the account ID for the currently logged-in user.
	runtime.envType	runtime.EnvType	Returns the current environment in which the script is executing.
	runtime.executionContext	enum	Returns a runtime.ContextType enumeration that represents what triggered the current script.
	runtime.queueCount	number (read-only)	Returns the number of scheduled script queues in a given account.
	runtime.version	string (read-only)	Returns 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 .
Enum	runtime.ContextType	enum	Enumeration that holds the context information about what triggered the current script. Returned by the <code>runtime.executionContext</code> property of the N/runtime Module .
	runtime.EnvType	enum	Enumeration that holds all possible environment types that the current script can execute in.
	runtime.Permission	enum	Enumeration that holds the user permission level for a specific permission ID. Returned by the

Member Type	Name	Return Type / Value Type	Description
			User.getPermission(options) method.

Script Object Members

Member Type	Name	Return Type / Value Type	Description
Method	Script.getParameter(options)	number Date string boolean	Returns the value of a script parameter for the currently executing script.
	Script.getRemainingUsage()	number	Returns a number value for the usage units remaining for the currently executing script.
Property	Script.deploymentId	string (read-only)	Returns the deployment ID for the script deployment on the currently executing script.
	Script.id	string (read-only)	Returns the script ID for the currently executing script.
	Script.logLevel	string (read-only)	Returns the script logging level for the current script execution. This method is not supported on client scripts.
	Script.percentComplete	number	Get or set the percent complete specified for the current scheduled script execution. The return value will appear in the % Complete column on the Scheduled Script Status page.
	Script.bundleIds	Array (read-only)	Returns an Array of bundle IDs for the bundles that include the currently executing script.

Session Object Members

Member Type	Name	Return Type / Value Type	Description
Method	Session.get(options)	string	Returns the user-defined session object value associated with the session object key.
	Session.set(options)	void	Sets a key and value for a user-defined session object.

User Object Members

Member Type	Name	Return Type / Value Type	Description
Method	User.getPermission(options)	number	Returns a user permission level for the specified permission as a runtime.Permission enumeration.
	User.getPreference(options)	string	Returns the value of a NetSuite preference. Currently only General Preferences and Accounting Preferences are exposed in SuiteScript. For more information about these preferences names and IDs, see the help topics General Preferences and Accounting Preferences .
Property	User.department	number (read-only)	Returns the internal ID of the department for the currently logged-in user.
	User.email	string (read-only)	Returns the email address of the currently logged-in user.
	User.id	number (read-only)	Returns the internal ID of the currently logged-in user.
	User.location	number (read-only)	Returns the internal ID of the location of the currently logged-in user.
	User.name	string (read-only)	Returns the name of the currently logged-in user.
	User.role	number (read-only)	Return the internal ID of the role for the currently logged-in user.
	User.roleCenter	string (read-only)	Returns the internal ID of the center type, or role center, for the currently logged-in user.
	User.roleId	string (read-only)	Returns the custom scriptId of the role for the currently logged-in user.
	User.subsidiary	number (read-only)	Returns the internal ID of the subsidiary for the currently logged-in user.

N/runtime Module Script Sample

i Note: These samples use the `define` function. The NetSuite Debugger cannot step though a `define` function. If you need to step through your code in the NetSuite Debugger, you must use a `require` function.

The following Suitelet sample writes user and session information for the currently executing script to the response.

```
/**
```

```
*@NApiVersion 2.x
*@NScriptType Suitelet
*/
define(['N/runtime'],
  function(runtime) {
    function onRequest(context) {
      var remainingUsage = runtime.getCurrentScript().getRemainingUsage();
      var userRole = runtime.getCurrentUser().role;
      runtime.getCurrentSession().set({
        name: 'scope',
        value: 'global'
      });
      var sessionScope = runtime.getCurrentSession().get({
        name: 'scope'
      });
      log.debug('Remaining Usage:', remainingUsage);
      log.debug('Role:', userRole);
      log.debug('Session Scope:', sessionScope);
      context.response.write('Executing under role: ' + userRole
        + '. Session scope: ' + sessionScope + '.');
    }
    return {
      onRequest: onRequest
    };
  });
});
```

The following scheduled script creates sales records during runtime and logs the record creation progress.

```
/**
*@NApiVersion 2.0
*@NScriptType scheduledscript
*/
define(['N/runtime', 'N/record'],
function(runtime, record){
  return {
    execute: function(context) {
      var script = runtime.getCurrentScript();
      for (x=0; x<500; x++) {
        var rec = record.create({
          type: record.Type.SALES_ORDER
        });
        script.percentComplete = (x * 100)/500;
        log.debug({
          title: 'New Sales Orders',
          details: "Record creation progress: " + script.percentComplete + "%"
        });
      }
    }
  );
});
```

runtime.Script

Object Description	Encapsulates the runtime settings of the currently executing script. Use <code>runtime.getCurrentScript()</code> to return this object. For a complete list of this object's methods and properties, see Script Object Members .
Supported Script Types	Server-side scripts
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
log.debug('Script ID: ' + scriptObj.id);
...
//Add additional code
```

Script.getParameter(options)

Method Description	Returns the value of a script parameter for the currently executing script.
Returns	number Date string Array
Supported Script Types	Server-side scripts
Governance	None
Module	N/runtime Module
Since	Version 2015 Release 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.	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
log.debug("Script parameter of custscript1: " +
    scriptObj.getParameter({name: 'custscript1'}));
...
//Add additional code
```

Script.getRemainingUsage()

Method Description	Returns a number value for the usage units remaining for the currently executing script.
Returns	number
Supported Script Types	Server-side scripts
Governance	None
Module	N/runtime Module
Since	Version 2015 Release 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 Sample](#).

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
log.debug("Remaining governance units: " + scriptObj.getRemainingUsage());
...
//Add additional code
```

Script.deploymentId

Property Description	Returns the deployment ID for the script deployment on the currently executing script.
Type	string (read-only)
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
log.debug("Deployment Id: " + scriptObj.deploymentId);
...
//Add additional code
```

Script.id

Property Description	Returns the script ID for the currently executing script.
Type	string (read-only)
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
log.debug("Script Id: " + scriptObj.id);
...
//Add additional code
```

Script.logLevel

Property Description	Returns the script logging level for the current script execution. This method is not supported on client scripts. Returns one of the following values:
Type	string (read-only)
Module	N/runtime Module

Since	Version 2015 Release 2
-------	------------------------

Syntax

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

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
log.debug("Logging level: " + scriptObj.logLevel);
...
//Add additional code
```

Script.percentComplete

Property Description	Get or set the percent complete specified for the current scheduled script execution. The return value appears in the % Complete column on the Scheduled Script Status page.
-----------------------------	---

Important: This property throws SSS_OPERATION_UNAVAILABLE if the currently executing script is not a scheduled script.

Type	number
Module	N/runtime Module
Since	Version 2015 Release 2

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 Sample](#).

```
//Gets the percentage of records completed
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
if (scriptObj.executionContext == ContextType.SCHEDULED)
{
    log.debug({
        details: "Script percent complete: " + scriptObj.percentComplete
    });
    ...
}
...
```

```
//Sets the percent complete
...
var script = runtime.getCurrentScript();
```

```

for (x=0; x<500; x++) {
    var rec = record.create({
        type:record.Type.SALES_ORDER
    });
    script.percentComplete = (x * 100)/500;
    log.debug({
        title: 'New Sales Orders',
        details: "Record creation progress: " + script.percentComplete + "%"
    });
}
...
//Add additional code

```

Script.bundleIds

Property Description	Returns an Array of bundle IDs for the bundles that include the currently executing script.
Type	Array (read-only)
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax



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

```

//Add additional code
...
var scriptObj = runtime.getCurrentScript();
var bundleArr = scriptObj.bundleIds;
...
//Add additional code

```

runtim.Session

Object Description	Encapsulates the user session for the currently executing script. Use this object to set and get user-defined objects for the current user session. Use the objects to track user-related session data. For example, you can gather information about the user scope, budget, or business problems. Use Session.set(options) to set session object values and then use Session.get(options) to retrieve the values. For a complete list of this object's methods, see Session Object Members .
Supported Script Types	Server-side scripts
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
var sessionObj = runtime.getCurrentSession();
sessionObj.set({name: "myKey", value: "myValue"});
log.debug("Session object myKey value: " + sessionObj.get({name: "myKey"}));
...
//Add additional code
```

Session.get(options)

Method Description	Returns the user-defined session object value associated with the session object key.
Returns	string
Supported Script Types	Server-side scripts
Governance	None
Module	N/runtime Module
Since	Version 2015 Release 2

Parameters



Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.name	string	Required	String used as a key to store the <code>runtime.Session</code> .	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
var sessionObj = runtime.getCurrentSession();
sessionObj.set({name: "myKey", value: "myValue"});
log.debug("Session object myKey value: " + sessionObj.get({name: "myKey"}));
...
//Add additional code
```

Session.set(options)

Method Description	Sets a key and value for a user-defined <code>runtime.Session</code> . Use <code>Session.get(options)</code> to retrieve the object value after you set it.
Returns	<code>void</code>
Supported Script Types	Server-side scripts
Governance	None
Module	N/runtime Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.name</code>	<code>string</code>	Required	Key used to store the <code>runtime.Session</code> .	Version 2015 Release 2
<code>options.value</code>	<code>string</code>	Required	Value to associate with the key in the user session.	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var sessionObj = runtime.getCurrentSession();
sessionObj.set({
    name: "myKey",
    value: "myValue"
});
log.debug("Session object myKey value: " + sessionObj.get({name: "myKey"}));
...
//Add additional code
```

runtime.User

Object Description	Encapsulates the properties and preferences for the user of the currently executing script. For a complete list of this object's methods and properties, see User Object Members .
---------------------------	--

Supported Script Types	Server-side scripts
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
...
//Add additional code
```

User.getPermission(options)

Method Description	Returns a user permission level for the specified permission as a <code>runtime.Permission</code> enumeration.
Returns	string
Supported Script Types	Server-side scripts
Governance	None
Module	N/runtime Module
Since	Version 2015 Release 2

Parameters

 **Note:** 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 <i>Permission Names and IDs</i> .	Version 2015 Release 2

Syntax

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

```
//Add additional code
```

```

...
var userObj = runtime.getCurrentUser();
log.debug("User permission of ADMI_ACCOUNTING:" +
    (userObj.getPermission('ADMI_ACCOUNTING') ==
        runtime.Permission.FULL?'FULL':userObj.getPermission('ADMI_ACCOUNTIN
G'));
...
//Add additional code

```

User.getPreference(options)

Method Description	Returns the value of a NetSuite preference. Currently only General Preferences and Accounting Preferences are exposed in SuiteScript. For more information about these preferences names and IDs, see the help topics General Preferences and Accounting Preferences . You can also view General Preferences by going to Setup > Company > General Preferences. View Accounting Preferences by going to Setup > Accounting > Accounting Preferences. If you want to change the value of a General or Accounting preference using SuiteScript 2.0, you must load each preference page using <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.Record</code> . You can use the <code>Record.setValue(options)</code> method to set the preference.
Returns	string
Supported Script Types	Server-side scripts
Governance	None
Module	N/runtime Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.name</code>	string	Required	Internal ID of the preference. For a list of preference IDs, see <i>Preference Names and IDs</i> .	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("User preference for emailemployeeonapproval: " + userObj.getPreference({name: "email
employeeonapproval"}));
...
//Add additional code
```

User.department

Property Description	Returns the internal ID of the department for the currently logged-in user.
Type	number (read-only)
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user department: " + userObj.department);
...
//Add additional code
```

User.email

Property Description	Returns the email address of the currently logged-in user. To use this property, the email field on the user employee record must contain an email address.
Note:	In a shopping context where the shopper is recognized but not logged in, this method can be used to return the shopper's email, instead of getting it from the customer record.
Type	string (read-only)
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Current user email: " + userObj.email);
...
//Add additional code
```

User.id

Property Description	Returns the internal ID of the currently logged-in user.
Type	number (read-only)
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user: " + userObj.id);
...
//Add additional code
```

User.location

Property Description	Returns the internal ID of the location of the currently logged-in user.
Type	number (read-only)
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user location: " + userObj.location);
```

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

User.name

Property Description	Returns the name of the currently logged-in user.
Type	string (read-only)
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Name of current user: " + userObj.name);
...
//Add additional code
```

User.role

Property Description	Return the internal ID of the role for the currently logged-in user.
Type	number (read-only)
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user role: " + userObj.role);
...
//Add additional code
```

User.roleCenter

Property Description	Returns the internal ID of the center type, or role center, for the currently logged-in user.
-----------------------------	---

Type	number (read-only)
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user center type (role center): " + userObj.roleCenter);
...
//Add additional code
```

User.roleId

Property Description	Returns the custom scriptId of the role for the currently logged-in user. You can use this value instead of the internal ID for the role. When bundling a custom role, the internal ID number of the role in the target account can change after the bundle is installed. Therefore, in the target account you can use this property to return the unique/custom scriptId assigned to the role.
Type	string
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Custom script ID of current user role: " + userObj.roleId);
...
//Add additional code
```

User.subsidiary

Property Description	Returns the internal ID of the subsidiary for the currently logged-in user.
Type	number (read-only)

Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user subsidiary: " + userObj.subsidiary);
...
//Add additional code
```

runtime.getCurrentScript()

Method Description	Returns a runtime.Script that represents the currently executing script. Use this method to get properties and parameters of the currently executing script and script deployment. If you want to get properties for the session or user, use runtime.getCurrentSession() or runtime.getCurrentUser() instead.
Returns	runtime.Script
Supported Script Types	Server-side scripts
Governance	None
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
...
//Add additional code
```

runtime.getCurrentSession()

Method Description	Returns a runtime.Session that represents the user session for the currently executing script.
---------------------------	--

	Use this method to get session objects for the current user session. If you want to get properties for the script or user, use runtime.getCurrentScript() or runtime.getCurrentUser() instead.
Returns	<code>runtime.Session</code>
Supported Script Types	Server-side scripts
Governance	None
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var sessionObj = runtime.getCurrentSession();
...
//Add additional code
```

runtime.getCurrentUser()

Method Description	Returns a <code>runtime.User</code> that represents the properties and preferences for the user of the currently executing script. Use this method to get session objects for the current user session. If you want to get properties for the script or session, use runtime.getCurrentScript() or runtime.getCurrentSession() instead.
Returns	<code>runtime.User</code>
Supported Script Types	Server-side scripts
Governance	None
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var userObj = runtime.getCurrentUser();
...
//Add additional code
```

runtime.isFeatureInEffect(options)

Method Description	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.
Returns	boolean true false
Supported Script Types	Server-side scripts
Governance	None
Module	N/runtime Module
Since	Version 2015 Release 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 Feature Names and IDs .	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
log.debug('Advanced Billing feature is enabled: ' + runtime.isFeatureInEffect({feature: "ADVBILLING"}));
...
//Add additional code
```

runtime.accountId

Property Description	Returns the account ID for the currently logged-in user.
Type	string (read-only)
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
log.debug("Account ID for the current user: " + runtime.accountId);
...
//Add additional code
```

runtime.envType

Property Description	Returns the current environment in which the script is executing. This property returns one of the values from the runtime.EnvType enumeration.
Type	runtime.EnvType
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
log.debug("Environment for current user: " + JSON.stringify(runtime.envType));
...
//Add additional code
```

runtime.executionContext

Property Description	Property that describes what triggered the current script. This value is set by the runtime.ContextType enumeration.
Type	runtime.ContextType
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
```

```

...
if (runtime.executionContext === runtime.ContextType.USEREVENT)
    return;
...
//Add additional code

```

runtim.queueCount

Property Description	Returns the number of scheduled script queues in a given account. This property is helpful for SuiteApp developers who want to check for the number of queues in an account. If the consumer of the SuiteApp has purchased a SuiteCloud Plus license, runtime.queueCount returns 5, meaning that the account has 5 scheduled script queues. The runtime.queueCount property in accounts that do not have a SuiteCloud Plus licence will return 1, meaning the account has only 1 scheduled script queue. In some cases, an account may have two licenses supporting 10 queues, or three licenses supporting 15 queues. Once you get the number of queues, you can make business logic decisions based on the number. For example, if you know an account has 5 queues, you can have more than 1 script deployment and distribute the processing load to more than 1 queue.
Type	number (read-only)
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax



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

```

//Add additional code
...
log.debug("Number of queues available: " + runtime.queueCount);
...
//Add additional code

```

runtim.version

Property Description	Returns the version of NetSuite that the method is called in. For example, the runtime.version property in an account running NetSuite 2015.2 is 2015.2. Use this method, for example, when installing a bundle in another NetSuite accounts and you want to know the version number before installing the bundle.
Type	string (read-only)
Module	N/runtime Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
log.debug("Current NetSuite version: " + runtime.version);
...
//Add additional code
```

runtime.ContextType

Enum Description	Enumeration that holds the context information about what triggered the current script. Returned by the <code>runtime.executionContext</code> property of the N/runtime Module .
Module	N/runtime Module
Since	Version 2015 Release 2

Values

<ul style="list-style-type: none"> ■ ACTION ■ BUNDLE_INSTALLATION ■ CONSOLRATEADJUSTOR ■ CSV_IMPORT ■ CUSTOMGLLINES ■ CUSTOM_MASSUPDATE ■ EMAIL_CAPTURE ■ MAP_REDUCE ■ PAYMENTGATEWAY 	<ul style="list-style-type: none"> ■ PORTLET ■ PROMOTIONS ■ SCHEDULED ■ SHIPPING_PARTNERS ■ SUITELET ■ TAX_CALCULATION ■ USEREVENT ■ USER_INTERFACE 	<ul style="list-style-type: none"> ■ WEBAPPLICATION ■ WEBSERVICES ■ WEBSTORE ■ WORKFLOW
--	---	---

runtime.EnvType

Enum Description	Enumeration that holds all possible environment types that the current script can execute in. One of these values is returned by the <code>runtime.envType</code> property.
-------------------------	---

	<p>Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Module	N/runtime Module
Since	Version 2015 Release 2

Values

- SANDBOX
- PRODUCTION
- BETA
- INTERNAL

runtime.Permission

Enum Description	<p>Enumeration that holds the user permission level for a specific permission ID. Returned by the User.getPermission(options) method. For information on working with NetSuite permissions, see the help topic NetSuite Permissions Overview.</p> <p>Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Module	N/runtime Module
Since	Version 2015 Release 2

Values

- FULL
- EDIT
- CREATE
- VIEW
- NONE

N/search Module

Load the search module to create and run ad-hoc or saved searches and analyze and iterate through the search results. You can search for a single record by keywords, create saved searches, search for duplicate records, or return a set of records that match filters you define.

You also have the option to paginate search results and construct navigation that jumps between the next and previous pages. Due to the performance benefits, this is a suitable approach for working with a large result set.

- [N/search Module Members](#)
- [Search Object Members](#)
- [Result Object Members](#)
- [Column Object Members](#)
- [Filter Object Members](#)
- [Page Object Members](#)
- [PagedData Object Members](#)
- [PageRange Object Members](#)
- [ResultSet Object Members](#)
- [N/search Module Script Samples](#)

N/search Module Members

Member Type	Name	Return Type / Value Type	Description
Object	search.Search	Object	Encapsulates a NetSuite search. Use the methods available to the Search object to create a search, run a search, or save a search.
	search.Result	Object	Encapsulate a single search result row. Use the methods and properties for the Result object to get the column values for the result row.
	search.Column	Object	Encapsulates a single search column in a <code>search.Search</code> object. Use the methods and properties available to the Column object to get or set Column properties.
	search.Filter	Object	Encapsulates a search filter used in a search. Use the properties for the Filter object to get and set the filter properties.
	search.ResultSet	Object	Encapsulates a set of search results returned by <code>Search.run()</code> .
	search.Page	Object	Encapsulates a set of search results for a single search page.
	search.PagedData	Object	Holds metadata about a paginated query.
	search.PageRange	Object	Defines the page range to bound the result set for a paginated query.
Method	search.create(options)	search.Search	Creates a new search and returns it as a <code>search.Search</code> object.

Member Type	Name	Return Type / Value Type	Description
	search.create.promise(options)	search.Search	Creates a new search asynchronously and returns it as a search.Search object.
	search.load(options)	search.Search	Loads an existing saved search and returns it as a search.Search object.
	search.load.promise(options)	search.Search	Loads an existing saved search asynchronously and returns it as a search.Search object.
	search.delete(options)	void	Deletes an existing saved search asynchronously and returns it as a search.Search object.
	search.delete.promise(options)	void	Deletes an existing saved search and returns it as a search.Search object.
	search.duplicates(options)	search.Result[]	Performs a search for duplicate records based on the duplicate detection configuration for the account. Returns an array of search.Result objects.
	search.duplicates.promise(options)	search.Result[]	Performs a search for duplicate records asynchronously based on the duplicate detection configuration for the account. Returns an array of search.Result objects.
	search.global(options)	search.Result[]	Performs a global search against a single keyword or multiple keywords.
	search.global.promise(options)	search.Result[]	Performs a global search asynchronously against a single keyword or multiple keywords.
	search.lookupFields(options)	Object array	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	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)	search.Column	Creates a new search column as a search.Column object.

Member Type	Name	Return Type / Value Type	Description
	search.createFilter(options)	search.Filter	Creates a new search filter as a search.Filter object.
Enum	search.Operator	enum	Enumeration that holds the values for search operators to use with the search.Filter object.
	search.Sort	enum	Enumeration that holds the values for supported sorting directions used with search.createColumn(options) .
	search.Summary	enum	Enumeration that holds the values for summary types used by the Column.summary object.
	search.Type	enum	Enumeration that holds the string values for record types that support search.create(options) .

Search Object Members

The following members are called on [search.Search](#).

Member Type	Name	Return Type / Value Type	Description
Method	Search.save()	number	Saves a search created by search.create(options) or loaded with search.load(options) . Returns the internal ID of the saved search.
	Search.save.promise()	number	Asynchronously saves a search created by search.create(options) or loaded with search.load(options) . Returns the internal ID of the saved search.
	Search.run()	search.ResultSet	Runs an ad-hoc search created with search.create(options) or a search loaded with search.load(options) , returning the results as a search.ResultSet .
	Search.runPaged(options)	search.PagedData	Runs the current search and returns a search.PagedData Object.
	Search.runPaged.promise(options)	search.PagedData	Asynchronously runs the current search and returns a search.PagedData Object.

Member Type	Name	Return Type / Value Type	Description
Property	Search.searchType	string	Internal ID name of the record type on which a search is based.
	Search.searchId	number	Internal ID of a search.
	Search.filters	search.Filter[]	Filters for the search as an array of search.Filter objects.
	Search.filterExpression	Object[]	Search filter expression for the search as an array of expression objects.
	Search.columns	search.Column[] string[]	Columns to return for this search as an array of search.Column objects or a string array of column names.
	Search.title	string	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	Script ID for a saved search, starting with <code>customsearch</code> .
	Search.isPublic	boolean <code>true</code> <code>false</code>	Value is <code>true</code> if the search is public, or <code>false</code> if it is not.

Column Object Members

The following members are called on [search.Column](#).

Member Type	Name	Return Type / Value Type	Description
Method	Column.setWhenOrderedBy(options)	search.Column	Returns the search column for which the minimal or maximal value should be found when returning the search.Column value.
Property	Column.name	string (read-only)	Name of a search column as a string.
	Column.join	string (read-only)	Join ID for a search column as a string.
	Column.summary	string (read-only)	Returns the summary type for a search column.
	Column.formula	string	Formula used for a search column as a string.
	Column.label	string	Label used for the search column. You can only get or set custom labels with this property.

Member Type	Name	Return Type / Value Type	Description
	Column.function	string	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	Description
Property	Filter.name	string (read-only)	Name or internal ID of the search field.
	Filter.join	string (read-only)	Join ID for the search filter.
	Filter.operator	string (read-only)	Operator used for the search filter.
	Filter.summary	search.Summary	Summary type for the search filter.
	Filter.formula	string	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	Description
Method	Page.next()	void	Gets the next segment of data from a paginated search
	Page.next.promise()	void	Asynchronously gets the next segment of data from a paginated search
	Page.prev()	void	Gets the previous segment of data from a paginated search
	Page.prev.promise()	void	Asynchronously gets the previous segment of data from a paginated search
Property	Page.data	search.Result[]	The results from a paginated search.
	Page.isFirst	read-only boolean	Indicates whether a page is the first page of data for a result set
	Page.isLast	read-only boolean	Indicates whether a page is the last page of data for a result set.

Member Type	Name	Return Type / Value Type	Description
	Page.pagedData	read-only search.PagedData	The PagedData Object used to fetch this Page Object.
	Page.pageRange	read-only search.PageRange	The PageRange Object used to fetch this Page Object.

PagedData Object Members

The following members are called on [search.PagedData](#).

Member Type	Name	Return Type / Value Type	Description
Method	PagedData.fetch(options)	void	Retrieves the data within the specified page range.
	PagedData.fetch.promise()	void	Asynchronously retrieves the data within the specified page range.
Property	PagedData.count	read-only number	The total number of results when Search.runPaged(options) was executed.
	PagedData.pageRanges	read-only search.PageRange[]	The collection of PageRange objects that divide the entire result set into smaller groups.
	PagedData.pageSize	read-only number	The maximum number of entries per page
	PagedData.searchDefinition	read-only search.Search	The search criteria used when Search.runPaged(options) was executed.

PageRange Object Members

The following members are called on [search.PageRange](#).

Member Type	Name	Return Type / Value Type	Description
Property	PageRange.compoundLabel	read-only string	Human-readable label with beginning and ending range identifiers
	PageRange.index	read-only number	The index of this page range.

Result Object Members

The following members are called on [search.Result](#).

Member Type	Name	Return Type / Value Type	Description
Method	Result.getValue(options)	string	Used on formula fields and non-formula (standard) fields to get the value of a specified search return column.
	Result.getValue(column)	string	Encapsulate a single search result row. Use the methods and properties for the search.Result to get the column values for the result row.
	Result.getText(column)	string	The text value for a search.Column if it is a stored select field.
	Result.getText(options)	string	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	Result.recordType	string (read-only)	The type of record returned in a search result row.
	Result.id	number(read-only)	The internal ID for the record returned in a search result row.
	Result.columns	search.Column[]	Array of search.Column 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	Description
Method	ResultSet.getRange(options)	search.Result[]	Retrieve a slice of the search result as an array of search.Result objects.
	ResultSet.getRange.promise(options)	search.Result[]	Asynchronously retrieve a slice of the search result as an array of search.Result objects.
	ResultSet.each(callback)	void	Use a developer-defined function to invoke on each row in the search results, up to 4000 results at a time.

Member Type	Name	Return Type / Value Type	Description
	ResultSet.each.promise(callback)	void	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[]	An array of <code>search.Column</code> objects that represent the columns returned in the search results.

N/search Module Script Samples

In your NetSuite account, the One World feature needs to be enabled in the account for the samples to work. These samples are designed to run from a OneWorld account.

Note: These sample scripts use the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see [SuiteScript 2.0 – Script Architecture](#) and [SuiteScript 2.0 Script Types and Entry Points](#).

The following examples create a saved search on the sales order record.

```
/**
 *@NApiVersion 2.x
 */
// This example creates a saved search on the salesorder record
require(['N/search'],
    function(search) {
        function createSearch() {
            var mySalesOrderSearch = search.create({
                type: search.Type.SALES_ORDER,
                title: 'My SalesOrder Search',
                id: 'customsearch_my_so_search',
                columns: ['entity', 'subsidiary', 'name', 'currency'],
                filters: [
                    ['mainline', 'is', 'T'],
                    'and', ['subsidiary.name', 'contains', 'CAD']
                ]
            });
            mySalesOrderSearch.save();
        }
        createSearch();
    });

```

```
/**
 *@NApiVersion 2.x
 */
require(['N/search'],
    function(search){
        function createAnotherSearch() {
            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']
        }]
    );
    mySalesOrderSearch.save();
}
createAnotherSearch();
});

```

The following example loads and runs a search on the sales order record, and uses a callback function on 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();
    });

```

The following example loads and runs a search on the sales order record, and gets the first 100 rows of 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();
}
);

```

The following example loads and runs a search on the sales order record, and uses a callback function on 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();
    });

```

The following example deletes a saved search.

```

/**
 * @NApiVersion 2.x
 */
require(['N/search'],
    function(search) {

```

```

function deleteSearch() {
    search.delete({
        id: 'customsearch_my_so_search'
    });
    deleteSearch();
}

```

search.Search

Object Description	Encapsulates a NetSuite search. Use the methods available to search.Search to create a search, run a search, or save a search.
	<p>Note: You do not need to save the search to run it.</p> <p>For more information about executing NetSuite searches using SuiteScript, see Searching Overview. For a complete list of this object's methods and properties, see Search Object Members.</p>
Supported Script Types	All script types
Module	N/search Module
Since	Version 2015 Release 2

Syntax

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()

Method Description	Runs an ad-hoc search created with search.create(options) or a search loaded with search.load(options) , returning the results as a search.ResultSet . Calling this method does not save the search. Use this method with search.create(options) to create and run ad-hoc searches that are never saved to the database. After you run a search, you can use ResultSet.each(callback) to iterate through the result set and process each result.
Returns	search.ResultSet
Supported Script Types	All script types

Governance	None
Module	N/search Module
Since	Version 2015 Release 2

Syntax

⚠ 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)

Method Description	Runs the current search and returns summary information about paginated results. Calling this method does not give you the result set or save the search. To retrieve data, use PagedData.fetch(options) .
Returns	search.PagedData
Supported Script Types	All script types
Governance	5 units
Module	N/search Module
Since	Version 2016 Release 1

Parameters

i Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.pageSize	number	optional	Maximum number of entries per page There is an upper limit, a lower limit, and a default setting:

Parameter	Type	Required / Optional	Description
			<ul style="list-style-type: none"> ■ The maximum number allowed is 1000. ■ The minimum number allowed is 5. ■ By default, the page size is set to 50 entries per 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
...
var pagedData = mySearch.runPaged({
  pageSize:50
});
...
//Add additional code
```

Search.runPaged.promise(options)

Method Description	Runs the current search asynchronously and returns a search.PagedData Object.
	 Note: For more information about using this method, see Search.runPaged(options) . For additional information on promises, see Promise object .
Returns	search.PagedData
Synchronous Version	Search.runPaged(options)
Supported Script Types	All client-side scripts
Governance	5 units
Module	N/search Module
Since	Version 2016 Release 1

Syntax



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

```
//Add additional code
...
mySearch.runPaged.promise().then(getPageRangesPromiseChain);
...
```

```
//Add additional code
```

Search.save()

Method Description	Saves a search created by search.create(options) or loaded with search.load(options) . 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 search.create(options) or by setting the Search.title and Search.id properties. If you do not set the saved search ID, NetSuite generates one for you. See Search.id .
	<p>Note: You do not need to set these properties if you load a previously saved search with search.load(options) and then save it.</p> <p>This method also includes a promise version, Search.save.promise(). For more information about promises, see Promise object.</p>
Returns	the internal search ID of the saved search as a number
Supported Script Types	All script types
Governance	5 units
Module	N/search Module
Since	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	{1}: Missing a required argument: {2}	Required Search.title property not set on search.Search .
NAME_ALREADY_IN_USE	A search has already been saved with that name. Please use a different name.	The Search.title property on search.Search 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 search.load() .	The Search.id property on search.Search 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()

Method Description	Asynchronously saves a search created by search.create(options) or loaded with search.load(options) . Returns the internal ID of the saved search.
	<p> Note: For more information about using this method, see Search.save(). For additional information on promises, see Promise object.</p>
Returns	number
Synchronous Version	Search.save()
Supported Script Types	All client-side scripts
Governance	5 units
Module	N/search Module
Since	Version 2015 Release 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
...
search.create.promise({
    type: search.Type.SALES_ORDER
})
.then(function(searchObj) {
    return searchObj.save.promise()
})
.then(function (result) {
    log.debug({
        details: "Completed: " + result
    });
    // do something after completion
})
.catch(function onRejected(reason) {
    // do something on rejection
});
...
//Add additional code
```

Search.searchType

Property Description	Internal ID name of the record type on which a search is based. Use this if you have the internal ID of the search, but do not know the record type the search was based on. For example, if the search was on a Customer record, this property is <code>customer</code> ; if the search was on the Sales Order record type, this property is <code>salesorder</code> .
-----------------------------	--

Type	read-only string
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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

Property Description	Internal ID of the search. The internal ID is available only when the search is either loaded with search.load(options) or after it has been saved with Search.save() . Typical values are 55 or 234 or 87, not a value like customsearch_mysearch. Any ID prefixed with customsearch is a script ID, not the internal system ID for a search.
Type	number
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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

Property Description	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 <code>search.Filter</code> objects to overwrite any prior filters. Use <code>null</code> to set an empty array and remove any existing filters on this search. Use <code>search.createFilter(options)</code> to create a filter.
Type	<code>search.Filter[]</code>
Module	N/search Module
Since	Version 2015 Release 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

Property Description	Search filter expression for the search as an array of expression objects. A search filter expression is a JavaScript string array of zero or more elements. Each element is one of the following:
-----------------------------	--

	<ul style="list-style-type: none"> ■ Operator - either 'NOT', 'AND', or 'OR' ■ Filter term ■ Nested search filter expression <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> <div style="background-color: #e0f2ff; border: 1px solid #d9e1f2; padding: 5px; margin-top: 10px;"> Note: If you want to get or set a search filters, use the <code>Search.filters</code> property. </div> <p style="margin-top: 10px;">For more information, see <i>Search Filter Expression Overview</i>.</p>
Type	Object[]
Module	N/search Module
Since	Version 2015 Release 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
...
function createSearch() {
    var mySalesOrderSearch = search.create({
        type: search.Type.SALES_ORDER,
        filters: [
            ['mainline', 'is', 'T'],
            'and', ['subsidiary.name', 'contains', 'CAD']
        ]
    });
...
//Add additional code
```

Search.columns

Property Description	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 search.Column objects or a single search.Column 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.
Type	<code>search.Column[] string[]</code>
Module	N/search Module
Since	Version 2015 Release 2

Errors

Error Code	Message	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.title

Property Description	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 <code>search.create(options)</code> . The <code>Search.title</code> property is required to save a search with Search.save() .
Type	<code>string</code>
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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

Property Description	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.
	Note: This is not the internal NetSuite ID for the saved search. See Search.searchId .
Type	number
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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 additonal code
...
function createSearch() {
    var mySalesOrderSearch = search.create({
        type: search.Type.SALES_ORDER,
        title: 'My SalesOrder Search',
        id: 'customsearch_my_so_search',
    });
...
//Add additional code

```

Search.isPublic

Property Description	Value is <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.
Type	<code>boolean true false</code>
Module	N/search Module

Since	Version 2015 Release 2
-------	------------------------

Syntax



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

Object Description	Encapsulate a single search result row. Use the methods and properties for search.Result to get the column values for the result row.
--------------------	---



Note: Use [search.ResultSet](#) for the set of results from a search.

For more information about executing NetSuite searches using SuiteScript, see [Searching Overview](#).

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

Supported Script Types	All script types
------------------------	------------------

Module	N/search Module
--------	-----------------

Since	Version 2015 Release 2
-------	------------------------

Syntax



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)

Method Description	Used on formula fields and non-formula (standard) fields to get the value of a specified search return column. For convenience, this method takes a single <code>search.Column</code> Object. The value of the specified result set column is returned.
Note:	This method is overloaded. You can also use <code>Result.getValue(options)</code> to get column values based on name, join and summary properties for a column.
Returns	string
Supported Script Types	All script types
Governance	None
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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];

var value = firstResult.getValue(resultSet.columns[1]); // get the value of the second column (zero-based index)
log.debug({
    title: 'Value:',
    details: value
});
...
//Add additional code

```

Result.getValue(options)

Method Description	Used on formula fields and non-formula (standard) fields to get the value of a specified search return column by specifying the name, join, and summary properties.
	<p>Note: This method is overloaded. You can also use <code>Result.getValue(column)</code> to get column values. For convenience, you can pass in a single <code>search.Column</code>.</p>
	<p>Important: If you have multiple search return columns and you apply grouping, all columns must include a summary property.</p>
Returns	Value of the search result column as a string
Supported Script Types	All script types
Governance	None
Module	N/search Module
Since	Version 2015 Release 2

Parameters

Note: The options parameter is a JavaScript object.			
Parameter	Type	Required / Optional	Description
options.name	string	Required	The search return column name.
options.join	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 Working with the SuiteScript Records Browser .
options.summary	search.Summary	Optional	The summary type for this column. See search.Summary .

Syntax

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].getValue({
        name: 'amount'
    });
    var entity = searchResults[i].getValue({
        name: 'name',
        join: 'location'
    });
    ...
    //Add additional code
}

```

Result.getText(column)

Method Description	The text value for a search.Column if it is a stored select field.
	<p>Note: This method is overloaded. You can also use Result.getText(column) to get column text value based on name, join and summary properties for a column.</p>
Returns	string
Supported Script Types	All script types
Governance	None
Module	N/search Module
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description
column	search.Column	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];

```

```

var value = firstResult.getText({
    column: resultSet.columns[1]
}); // get the text value of the second column (zero-based index)

log.debug({
    title: 'Value: ',
    details: value
});
...
//Add additional code

```

Result.getText(options)

Method Description	The UI display name, or text value, for a search result column.
	<p>Important: The following field types are supported: select, image, or document fields.</p> <p>Note: This method is overloaded. You can also use Result.getText(column) to get a column value. For convenience, you can pass in a single search.Column.</p>
Returns	string
Supported Script Types	All script types
Governance	None
Module	N/search Module
Since	Version 2015 Release 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.
options.summary	search.Summary	Optional	The summary type used for the search column. See search.Summary .

Syntax

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

Property Description	The type of record returned in a search result row.
Type	search.Type enum
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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

Property Description	The internal ID for the record returned in a search result row.
Type	number (read-only)
Module	N/search Module

Since	Version 2015 Release 2
-------	------------------------

Syntax



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();
resultSet.each(function(result) {
  log.debug({
    title: 'Record Internal ID: ',
    details: result.id
  });
  return true;
});

...
//Add additional code
```

Result.columns

Property Description	Array of <code>search.Column</code> objects that encapsulate the columns returned in the search result row.
Type	<code>search.Column[]</code>
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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

Object Description	Encapsulates a single search column in a search.Search . Use the methods and properties available to the Column object to get or set Column properties. You create a search column object with search.createColumn(options) and add it to a search.Search object that you create with search.create(options) or load with search.load(options) . You can pass a Column object as a parameter to the Result.getValue(column) or Result.getText(column) methods. In addition, search.ResultSet contains an array of Column objects returned in the results of a search. For a complete list of this object's methods and properties, see Column Object Members .
Supported Script Types	All script types
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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)

Method Description	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.
Returns	<code>search.Column</code>
Supported Script Types	All script types
Governance	None
Module	N/search Module
Since	Version 2015 Release 2

Parameters

Note: 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

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

```
//Add additional code
...
// Execute a customer search that returns the amount of the most recent sales order per customer

var filters = [];
var columns = [];
```

```

filters[0] = search.createFilter({
  name: 'recordtype',
  join: 'transaction',
  operator: search.Operator.IS,
  values: 'salesorder'
});
filters[1] = search.createFilter({
  name: 'mainline',
  join: 'transaction',
  operator: search.Operator.IS,
  values: true
});
columns[0] = search.createColumn({
  name: 'entityid',
  summary: search.Summary.GROUP
});
columns[1] = search.createColumn({
  name: 'totalamount',
  join: 'transaction',
  summary: search.Summary.MAX
});
columns[1].setWhenOrderedBy({
  name: 'trandate',
  join: 'transaction'
});
var mySearch = search.create({
  type: 'customer',
  filters: filters,
  columns: columns
});
var resultsArray = mySearch.run().getRange({
  start: 0,
  end: 100
});
...
//Add additional code

```

Column.name

Property Description	Name of a search column as a string.
Type	string (read-only)
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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 Column Name: '
        + columnObj.name
});
...
//Add additional code

```

Column.join

Property Description	Join ID for a search column as a string.
Type	string (read-only)
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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: 'Join ID for Search Column: '
        + columnObj.join
});
...
//Add additional code

```

Column.summary

Property Description	Returns the summary type for a search column.
Type	search.Summary enum
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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

Property Description	Formula used for a search column as a string. To set this value, you must use formulatext, formulanumeric, formuladatetime, formulapercent, or formulacurrency.
Type	string
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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

Property Description	Label used for the search column. You can only get or set custom labels with this property.
Type	string
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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

Property Description	Special function applied to values in a search column. See Supported Functions .
Type	string
Module	N/search Module
Since	Version 2015 Release 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	
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	
numberAsTime	Number as Time	No	text
quarter	Quarter	Yes	text
rank	Rank	No	integer
round	Round	No	

Internal ID	Name	Date Function	Output
roundToHundredths	Round to Hundredths	No	
roundToTenths	Round to Tenths	No	
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

Property Description	The sort order of the column. Use the search.Sort enum to set the value. If Column.sort is not set, the column is not sorted in any particular order.
Type	search.Sort enum
Module	N/search Module
Since	Version 2015 Release 2

Syntax

⚠ 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

Object Description	Encapsulates a search filter used in a search. Use the properties for the Filter object to get and set the filter properties. You create a search filter object with search.createFilter(options) and add it to a search.Search object that you create with search.create(options) or load with search.load(options) .
	<p>Note: 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"> ■ To search for a "none of null" value, meaning do not show results without a value for the specified field, use a value of @NONE@ in the Filter.formula property. ■ To search on checkbox fields, use the IS operator with a value of T or F to search for checked or unchecked fields, respectively. <p>For a complete list of this object's methods and properties, see Filter Object Members.</p>
Supported Script Types	All script types
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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

Property Description	Name or internal ID of the search field as a string.
-----------------------------	--

	For more information, see search.createFilter(options) .
Type	string (read-only)
Module	N/search Module
Since	Version 2015 Release 2

Syntax

⚠ 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

Property Description	Join ID for the search filter as a string.
Type	string (read-only)
Module	N/search Module
Since	Version 2015 Release 2

Syntax

⚠ 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: 'Join ID: '
        + filterObj.join
});
...
//Add additional code
```

Filter.operator

Property Description	Operator used for the search filter. See search.Operator .
Type	string (read-only)

Module	N/search Module
Since	Version 2015 Release 2

Syntax

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: 'Operator Used: '
        + filterObj.operator
});
...
//Add additional code
```

Filter.summary

Property Description	Summary type for the search filter. Use this property to get or set the value of the summary type. See search.Summary .
Type	search.Summary
Module	N/search Module
Since	Version 2015 Release 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',
    summary: search.Summary.GROUP
});
...
//Add additional code
```

Filter.formula

Property Description	Formula used by the search filter. Use this property to get or set the formula used by the search filter. For more information about the formula property, see search.createFilter(options) .
Type	string
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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

Object Description	Encapsulates a set of search results returned by Search.run() . Use the methods and properties for the ResultSet object to iterate through each result returned by the search or access an arbitrary slice of results, up to 1000 results at a time. For a complete list of this object's methods and properties, see ResultSet Object Members .
Supported Script Types	All script types
Module	N/search Module
Since	Version 2015 Release 2

Syntax



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.run().each(function(result) {
    var entity = result.getValue({
        name: 'entity'
    });
    var subsidiary = result.getValue({
        name: 'subsidiary'
    });
    return true;
});
...
//Add additional code

```

ResultSet.getRange(options)

Method Description	Retrieve a slice of the search result as an array of search.Result objects. The start parameter is the inclusive index of the first result to return. The end parameter is the exclusive index of the last result to return. For example, getRange(0, 10) retrieves 10 search results, at index 0 through index 9. Unlimited rows in the result are supported, however you can only return 1,000 at a time based on the index values. If there are fewer results available than requested, then the array will contain fewer than end - start entries. For example, if there are only 25 search results, then getRange(20, 30) will return an array of 5 search.Result objects.
Returns	search.Result[]
Supported Script Types	All script types
Governance	10 units
Module	N/search Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
options.start	number	Required	Index number of the first result to return, inclusive.
options.end	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)

Method Description	Method used to asynchronously retrieve a slice of the search result as an array of search.Result objects.
	<p>Note: For information about the parameters and errors thrown for this method, see ResultSet.getRange(options). For additional information on promises, see Promise object.</p>
Returns	search.Result[]
Supported Script Types	All client-side scripts
Governance	10 units
Module	N/search Module
Since	Version 2015 Release 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)

Method Description	Use a developer-defined function to invoke on each row in the search results, up to 4000 results at a time. The callback function must use the following signature: <code>boolean callback(result.Result result);</code> The callback function takes a <code>search.Result</code> 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> .
Returns	<code>void</code>
Supported Script Types	All script types
Governance	10 units
Module	N/search Module
Since	Version 2015 Release 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 <code>search.Result</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
...
mySearch.run().each(function(result) {
    var entity = result.getValue({
        name: 'entity'
    });
    var subsidiary = result.getValue({
```

```

        name: 'subsidiary'
    });
    return true;
});
...
//Add additional code

```

ResultSet.each.promise(callback)

Method Description	Asynchronously uses a developer-defined function to invoke on each row in the search results, up to 4000 results at a time. The callback function must use the following signature:
	<p> Note: For information about the parameters and errors thrown for this method, see ResultSet.each(callback). For additional information on promises, see Promise object.</p>
Returns	void
Supported Script Types	All client-side scripts
Governance	10 units
Module	N/search Module
Since	Version 2015 Release 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
...
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

Property Description	An array of search.Column objects that represent the columns returned in the search results.
Type	search.Column[] This property is read-only
Module	N/search Module
Since	Version 2015 Release 2

Syntax

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

Object Description	Encapsulates an individual search page containing a result set for a paginated search. For a complete list of this object's methods and properties, see Page Object Members .
Supported Script Types	All script types
Module	N/search Module

Since	Version 2015 Release 1
-------	------------------------

Syntax



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

```
//Add additional code
...
var page = pagedData.fetch({
    index: lastPageRange.index
});
...
//Add additional code
```

Page.next()

Method Description	Method used to fetch the next segment of data (bounded by search.PageRange). Moves the current page to next range.
Returns	Void
Supported Script Types	All script types
Governance	5 units
Module	N/search Module
Since	Version 2016 Release 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()

Method Description	Method used to asynchronously fetch the next segment of data (bounded by search.PageRange). Moves the current page to another range. The promise is complete when the data for this range is loaded or rejected.
Returns	Void
Synchronous Version	Page.next()
Supported Script Types	All client-side scripts
Governance	5 units
Module	N/search Module
Since	Version 2016 Release 1

Syntax



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

```
//Add additional code
...
return mypage.next.promise().then(processPage);
...
//Add additional code
```

Page.prev()

Method Description	Method used to fetch the previous segment of data (bounded by search.PageRange). Moves the current page to previous range.
Returns	Void
Supported Script Types	All script types
Governance	5 units
Module	N/search Module
Since	Version 2016 Release 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()

Method Description	<p>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.</p> <p>Note: For information about errors thrown for this method, see Page.prev(). For additional information on promises, see Promise object.</p>
Returns	Void
Synchronous Version	<code>Page.prev()</code>
Supported Script Types	All client-side scripts
Governance	5 units
Module	N/search Module
Since	Version 2016 Release 1

Syntax



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

```
//Add additional code
...
return mypage.prev.promise().then(processPage);
...
//Add additional code
```

Page.data

Property Description	The results from a paginated search.
Type	<code>search.Result[]</code> This property is read-only.

Module	N/search Module
Since	Version 2016 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
...
function processPage(page){
    page.data.forEach(function(value){
        log.debug({
            details: "data: " + page.data
        });
    ...
//Add additional code
```

Page.isFirst

Property Description	Indicates whether the page is within the first range of the result set. Flags the start of the data collection.
Type	boolean true false This property is read-only.
Module	N/search Module
Since	Version 2016 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
...
while (!page.isFirst){
    page = page.next();
}
//Add additional code
```

Page.isLast

Property Description	Indicates whether a page is within the last range of the result set. Flags the end of the data collection.
Type	boolean true false

	This property is read-only.
Module	N/search Module
Since	Version 2016 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
...
while (!page.isLast){
    page = page.prev();
}
//Add additional code
```

Page.pagedData

Property Description	The PagedData Object used to fetch this Page Object.
Type	search.PagedData This property is read-only.
Module	N/search Module
Since	Version 2016 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 lastPageRange = pagedData.pageRanges[pagedData.pageRanges.length - 1];
...
//Add additional code
```

Page.pageRange

Property Description	The PageRange Object used to fetch this Page Object. Page boundary information with the key and label.
Type	search.PageRange This property is read-only.
Module	N/search Module
Since	Version 2016 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
...
log.debug({
    details: "Page Range: " + mySearchPage.pageRange
});
...

//Add additional code
```

search.PagedData

Object Description	Holds metadata for a paginated query. This object provides a high-level view of a search result, giving the total count of records, a list of pages ranges, and page size. For a complete list of this object's methods and properties, see PagedData Object Members .
Supported Script Types	All script types
Module	N/search Module
Since	Version 2015 Release 1

Syntax



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

```
//Add additional code
...
var pagedData = mySearch.runPaged({
    pageSize:1000
});
...

//Add additional code
```

PagedData.fetch(options)

Method Description	This method retrieves the data within the specified page range. This method also includes a promise version, PagedData.fetch.promise(). For more information about promises, see Promise object .
Returns	search.Page
Supported Script Types	All script types

Governance	5 units
Module	N/search Module
Since	Version 2016 Release 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()

Method Description	This method asynchronously retrieves the data bounded by the pageRange parameter.
	 Note: For information about the parameters and errors thrown for this method, see PagedData.fetch(options) . For additional information on promises, see Promise object .
Returns	search.Page
Synchronous Version	PagedData.fetch(options)
Supported Script Types	All client-side scripts
Governance	5 units
Module	N/search Module

Since	Version 2016 Release 1
-------	------------------------

Syntax

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

```
//Add additional code
...
return pagedData.fetch.promise().then(processPage);
...
//Add additional code
```

PagedData.count

Property Description	The total number of results when <code>Search.runPaged(options)</code> was executed.
Type	<code>number</code> This property is read-only.
Module	N/search Module
Since	Version 2016 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](#).

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

Property Description	The collection of <code>PageRange</code> objects that divide the entire result set into smaller groups. Includes page range information with the key and label for rendering.
Type	<code>search.PageRange[]</code>

	This property is read-only.
Module	N/search Module
Since	Version 2016 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
...
log.debug({
    details: "PageRange Array: " + myPagedData.pageRanges
});
...
//Add additional code
```

PagedData.pageSize

Property Description	Maximum number of entries per page Possible values are 5 - 1000 entries per page.
Type	number This is a read-only property.
Module	N/search Module
Since	Version 2016 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
...
log.debug({
    details: "Max Page Size: " + myPagedData.pageSize
});
...
//Add additional code
```

PagedData.searchDefinition

Property Description	The search criteria used to execute the result set for this PagedData Object.
----------------------	---

Type	read-only <code>search.Search</code>
Module	N/search Module
Since	Version 2016 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
...
log.debug({
    details: "Search Details: " + myPagedData.searchDefinition
});
...
//Add additional code
```

search.PageRange

Object Description	Defines the page range to contain the result set For a complete list of this object's properties, see PageRange Object Members .
Supported Script Types	All script types
Module	N/search Module
Since	Version 2015 Release 1

Syntax



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

```
//Add additional code
...
var page = pagedData.fetch({
    index: lastPageRange
});
...
//Add additional code
```

PageRange.compoundLabel

Property Description	Human-readable label with beginning and ending range identifiers
Type	read-only string

Module	N/search Module
Since	Version 2016 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
...
log.debug({
    details: "Page Range Description: " + myPageRange.compoundLabel
});
...
//Add additional code
```

PageRange.index

Property Description	The index of the pageRange
Type	number This property is read-only.
Module	N/search Module
Since	Version 2016 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
...
log.debug({
    details: "Page Range Index: " + myPageRange.index
});
...
//Add additional code
```

search.create(options)

Method Description	Creates a new search and returns it as a <code>search.Search</code> object. The search can be modified and run as an ad-hoc search with <code>Search.run()</code> , without saving it. Alternatively, calling <code>Search.save()</code> will save the search to the database, so it can be reused later in the UI or loaded with <code>search.load(options)</code> .
---------------------------	---

	<p>Note: 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 Promise object.</p>
Returns	<code>search.Search</code>
Supported Script Types	All script types
Governance	None
Module	N/search Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.type</code>	<code>string</code>	Required	Record internal ID for the record type you want to search. Use the <code>search.Type</code> enum for this argument.	Version 2015 Release 2
<code>options.filters</code>	<code>search.Filter[] Object[]</code>	Optional	<p>A single <code>search.Filter</code> object, an array of <code>search.Filter</code> 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"> ■ name (required) ■ join ■ operator (required) ■ summary ■ formula <p>For more information about these properties, see Filter Object Members.</p> <p>Note: You can further filter the returned <code>search.Search</code> object by adding additional filters with <code>Search.filters</code> or <code>Search.filterExpression</code>.</p>	Version 2015 Release 2
<code>options.filterExpression</code>	<code>Object[]</code>	Optional	Search filter expression for the search as an array of expression objects.	2016.2

Parameter	Type	Required / Optional	Description	Since
			<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"> ■ Operator - either 'NOT', 'AND', or 'OR' ■ Filter term ■ Nested search filter expression <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> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Note: If you want to get or set a search filters, use the <code>Search.filters</code> property. </div> <p>This parameter sets the value for the <code>Search.filterExpression</code> property.</p>	
<code>options.columns</code>	<code>search.Column[] Object[]</code>	Optional	<p>A single <code>search.Column</code> object or array of <code>search.Column</code> objects. 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"> ■ name (required) ■ formula ■ function ■ join ■ label ■ sort ■ summary <p>For more information about these properties, see Column Object Members.</p>	Version 2015 Release 2
<code>options.title</code>	<code>string</code>	Optional	The name for a saved search. The title property is required to save a search with <code>Search.save()</code> .	Version 2015 Release 2
<code>options.id</code>	<code>string</code>	Optional	Script ID for a saved search. If you do not set the saved search ID, NetSuite generates one for you. See <code>Search.id</code> .	Version 2015 Release 2
<code>options.isPublic</code>	<code>boolean true false</code>	Optional	Set to <code>true</code> to make the search public. Otherwise, set to <code>false</code> .	2016.2

Parameter	Type	Required / Optional	Description	Since
			This parameter sets the value for the <code>Search.isPublic</code> property.	

Errors

Error Code	Message	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	{1}: Missing a required argument: {2}	Required parameter is missing.
<code>SSS_INVALID_SRCH_FILTER_EXPR</code>	Malformed search filter expression. This is a general error raised when a filter expression cannot be parsed. For example: [f1, 'and', 'and', f2]	The <code>options.filters</code> parameter is not a valid search filter, filter array, or filter expression.
<code>SSS_INVALID_SRCH_COLUMN</code>	An <code>search.Column</code> Object contains an invalid column, or is not in proper syntax: {1}.	The <code>options.columns</code> parameter is not a valid column, string, or column or string 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/search Module Script Samples](#).

```
//Add additional code
...
var mySalesOrderSearch = search.create({
    type: search.Type.SALES_ORDER,
    title: 'My SalesOrder Search',
    id: 'customsearch_my_so_search',
    columns: [
        {
            name: 'entity'
        },
        {
            name: 'subsidiary'
        },
        {
            name: 'name'
        },
        {
            name: 'currency'
        }
    ],
    filters: [
        {
            name: 'mainline',
            operator: 'is',
            values: ['T']
        },
        {
            name: 'subsidiary.name',
            operator: 'contains',
            values: ['CAD']
        }
    ]
}
...
//Add additional code
```

search.create.promise(options)

Method Description	Creates a new search asynchronously and returns it as a search.Search object.
	<p>Note: For information about the parameters and errors thrown for this method, see search.create(options). For additional information on promises, see Promise object.</p>
Returns	search.Search
Synchronous Version	search.create(options)
Supported Script Types	All client-side scripts
Governance	None
Module	N/search Module
Since	Version 2015 Release 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
...
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.load(options)

Method Description	Loads an existing saved search and returns it as a search.Search . The saved search could have been created using the UI or created with search.create(options) and Search.save() .
---------------------------	---

	The search.load(options) method also includes a promise version, search.load.promise(options). For more information about promises, see Promise object .
Returns	search.Search
Supported Script Types	All script types
Governance	5 units
Module	N/search Module
Since	Version 2015 Release 2

Parameters

 **Note:** 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 customsearch. See Search.id .	Version 2015 Release 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 options.id 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)

Method Description	Loads an existing saved search asynchronously and returns it as a <code>search.Search</code> object. The saved search could have been created using the UI or created with <code>search.create(options)</code> and <code>Search.save()</code> .
	<p>Note: For information about the parameters and errors thrown for this method, see <code>search.load(options)</code>. For additional information on promises, see Promise object.</p>
Returns	<code>search.Search</code>
Synchronous Version	<code>search.load(options)</code>
Supported Script Types	All client-side scripts
Governance	5 units
Module	N/search Module
Since	Version 2015 Release 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
...
search.load.promise({
    type : search.Type.SALES_ORDER,
    id : 'customsearch_txn_search_salesorder'
})
.then(function (result) {
    log.debug({
        details: "Completed: " + result
    });
    // do something after completion
})
.catch(function onRejected(reason) {
    // do something on rejection
});
...
//Add additional code
```

search.delete(options)

Method Description	Deletes an existing saved search. The saved search could have been created using the UI or created with <code>search.create(options)</code> and <code>Search.save()</code> . The <code>search.delete(options)</code> method also includes a promise version, <code>search.delete.promise(options)</code> . For more information about promises, see Promise object .
---------------------------	--

Returns	void
Supported Script Types	All script types
Governance	5 units
Module	N/search Module
Since	Version 2015 Release 2

Parameters

 **Note:** 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 Search.id .	Version 2015 Release 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

 **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.delete({
    id: 'customsearch_my_so_search'
});
...
//Add additional code
```

search.delete.promise(options)

Method Description	Deletes an existing saved search asynchronously and returns it as a <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> .
--------------------	--

	<p> Note: For information about the parameters and errors thrown for this method, see search.delete(options). For additional information on promises, see Promise object.</p>
Returns	void
Synchronous Version	search.delete(options)
Supported Script Types	All client-side scripts
Governance	5 units
Module	N/search Module
Since	Version 2015 Release 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
...
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)

Method Description	<p>Performs a search for duplicate records based on the account's duplicate detection configuration.</p> <p>The <code>search.duplicates(options)</code> method also includes a promise version, <code>search.duplicates.promise(options)</code>. For more information about promises, see Promise object.</p> <p> Important: This API is for only records that support duplicate record detection. For example, customers, leads, prospects, contacts, partners, and vendors records.</p> <p>For more information about duplicate record detection, see the help topic Duplicate Record Detection</p>
---------------------------	---

Returns	search.Result [] that correspond to the duplicate record Results are limited to 1000 rows If there are no search results, this method returns null.
Supported Script Types	All script types
Governance	10 units
Module	N/search Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.type	enum	Required	Record internal ID name for which you want to check for duplicates. Use the search.Type enum for this argument.	Version 2015 Release 2
options.fields	Object	Optional	A set of key/value pairs used to detect duplicates. For example, email:'sample@test.com'. The keys are internal ID names of the fields used to detect the duplicate. For example, use companyname email name phone address1 city state zipcode.	Version 2015 Release 2
options.id	number	Optional	Internal ID of an existing record.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
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 duplicatesRecords = search.duplicates({
```

```

        type: search.Type.CONACTS,
        id: 'customsearch_my_contacts_search'
    });
...
//Add additional code

```

search.duplicates.promise(options)

Method Description	Performs a search for duplicate records asynchronously based on the Duplicate Detection configuration for the account. Returns an array of search.Result objects. This method only applies to records that support duplicate record detection. These records include customer lead prospect partner vendor contact.
Returns	search.Result[]
Synchronous Version	search.duplicates(options)
Supported Script Types	All client-side scripts
Governance	10 units
Module	N/search Module
Since	Version 2015 Release 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
...
search.duplicates.promise({
    type: search.Type.CUSTOMER,
    id: 28
})
.then(function (result) {
    log.debug({
        details: "Completed: " + result
    });
    // do something after completion
})
.catch(function onRejected(reason) {
    // do something on rejection
});
...
//Add additional code

```

search.global(options)

Method Description	Performs a global search against a single keyword or multiple keywords. Similar to the global search functionality in the UI, you can programmatically filter the global search results that are returned. For example, you can use the following filter to limit the returned records to Customer records: <code>'cu: simpson'</code> The <code>search.global(options)</code> method also includes a promise version, <code>search.global.promise(options)</code> . For more information about promises, see Promise object . For more information about global search, see the help topic Global Search .
Returns	<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> .
Supported Script Types	All script types
Governance	10 units
Module	N/search Module
Since	Version 2015 Release 2

Parameters

Note: 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><th>Since</th></tr> </thead> <tbody> <tr> <td><code>options.keywords</code></td><td>string</td><td>Required</td><td>Global search keywords string or expression.</td><td>Version 2015 Release 2</td></tr> </tbody> </table>	Parameter	Type	Required / Optional	Description	Since	<code>options.keywords</code>	string	Required	Global search keywords string or expression.	Version 2015 Release 2
Parameter	Type	Required / Optional	Description	Since						
<code>options.keywords</code>	string	Required	Global search keywords string or expression.	Version 2015 Release 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)

Method Description	Performs a global search asynchronously against a single keyword or multiple keywords. Returns an array of search.Result objects with four columns: name , type , info1 , and info2 .
Returns	search.Result[]
Synchronous Version	search.global(options)
Supported Script Types	All client-side scripts
Governance	10 units
Module	N/search Module
Since	Version 2015 Release 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
...
search.global.promise({
    keywords: 'Alan Rath'
})
.then(function (result) {
    log.debug({
        details: "Completed: " + result
    });
    // do something after completion
})
.catch(function onRejected(reason) {
    // do something on rejection
});
...
//Add additional code
```

search.lookupFields(options)

Method Description	Performs a search for one or more body fields on a record. You can use joined-field lookups with this method, with the following syntax: <code>join_id.field_name</code>
---------------------------	--

	<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 Promise object.</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 1234. 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": "1,2", "text": "US Sub, EU Sub" }] }</pre>
Returns	<p>Object array</p> <ul style="list-style-type: none"> ■ Returns select fields as an object with value and text properties. ■ Returns multiselect fields as an array of object with value:text pairs.
Supported Script Types	All script types
Governance	1 unit
Module	N/search Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.type</code>	enum	Required	Record internal ID name for which you want to look up fields. Use the search.Type enum for this argument.	Version 2015 Release 2
<code>options.id</code>	string	Required	Internal ID for the record, for example 777 or 87.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.columns	string string[]	Required	Array of column/field names to look up, or a single column/field name. The <code>columns</code> parameter can also be set to reference joined fields.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
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)

Method Description	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.
	Note: For information about the parameters and errors thrown for this method, see search.lookupFields(options) . For additional information on promises, see Promise object .
Returns	object array
Synchronous Version	search.lookupFields(options)
Supported Script Types	All client-side scripts
Governance	1 unit
Module	N/search Module
Since	Version 2015 Release 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
...
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)

Method Description	Creates a new search column as a <code>search.Column</code> object.
Returns	<code>search.Column</code>
Supported Script Types	All script types
Governance	None
Module	N/search Module
Since	Version 2015 Release 2

Parameters

i Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.name	string	Required	Name of the search column. See <code>Column.name</code> .	Version 2015 Release 2
options.join	string	Optional	Join ID for the search column. See <code>Column.join</code> .	Version 2015 Release 2
options.summary	enum	Optional	Summary type for the column. See	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			search.Summary and Column.summary.	
options.formula	string	Optional	Formula for the search column. See Column.formula.	Version 2015 Release 2
options.function	string	Optional	Special function for the search column. See Column.function.	Version 2015 Release 2
options.label	string	Optional	Label for the search column. See Column.label.	Version 2015 Release 2
options.sort	enum	Optional	The sort order of the column. Use the search.Sort enum for this argument. Also see Column.sort.	Version 2015 Release 2

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 search.Column object contains an invalid column summary type, or is not in proper syntax: {1}.	The options.summary parameter is not a valid search summary type. See search.Summary.
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)

Method Description	Creates a new search filter as a search.Filter object.
Returns	search.Filter

Supported Script Types	All script types
Governance	None
Module	N/search Module
Since	Version 2015 Release 2

Parameters

i	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.	Version 2015 Release 2
options.join	string	Optional	Join ID for the search filter.	Version 2015 Release 2
options.operator	search.Operator	Required	Operator used for the search filter. Use the search.Operator enum.	Version 2015 Release 2
options.values	string Date number string[] Date[]	Optional	Values to be used as filter parameters.	Version 2015 Release 2
options.formula	string	Optional	Formula used by the search filter.	Version 2015 Release 2
options.summary	search.Summary	Optional	Summary type for the search filter. See search.Summary .	Version 2015 Release 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 search.Column 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 search.Summary .
SSS_INVALID_SRCH_OPERATOR	An search.Filter object contains an invalid operator, or is not in proper syntax: {1}.	options.operator parameter is not a valid operator type. See search.Operator .

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
...
var mySearchFilter = search.createFilter({
    name: 'entity',
    operator: search.Operator.ISEMPTY
});
...
//Add additional code
```

search.Operator

Enum Description	Enumeration that holds the values for search operators to use with the search.Filter . See the help topic Search Operators for more information about the field types supported for each operator type.
Module	N/search Module
Since	Version 2015 Release 2

Values

<ul style="list-style-type: none"> ■ AFTER ■ ALLOF ■ ANY ■ ANYOF ■ BEFORE ■ BETWEEN ■ CONTAINS ■ DOESNOTCONTAIN ■ DOESNOTSTARTWITH ■ EQUALTO ■ GREATERTHAN ■ GREATERTHANOREQUALTO ■ HASKEYWORDS 	<ul style="list-style-type: none"> ■ IS ■ ISEMPTY ■ ISNOT ■ ISNOTEMPTY ■ LESSTHAN ■ LESSTHANOREQUALTO ■ NONEOF ■ NOTAFTER ■ NOTALLOF ■ NOTBEFORE ■ NOTBETWEEN ■ NOTEQUALTO ■ NOTGREATERTHAN 	<ul style="list-style-type: none"> ■ NOTGREATERTHANOREQUALTO ■ NOTLESSTHAN ■ NOTLESSTHANOREQUALTO ■ NOTON ■ NOTONORAFTER ■ NOTONORBEFORE ■ NOTWITHIN ■ ON ■ ONORAFTER ■ ONORBEFORE ■ STARTSWITH ■ WITHIN
--	--	--

Syntax



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

```
//Add additional code...
var mySearchFilter = search.createFilter({
    name: 'entity',
    operator: search.Operator.ISEMPTY
});
...

//Add additional code
```

search.Sort

Enum Description	Enumeration that holds the values for supported sorting directions used with search.createColumn(options) .
Module	N/search Module
Since	Version 2015 Release 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

Enum Description	Enumeration that holds the values for summary types used by the Column.summary or Filter.summary properties. For more information about each summary type, see the help topic Search Summary Types .
Module	N/search Module
Since	Version 2015 Release 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

Enum Description	Enumeration that holds the string values for record types that support search <code>search.create(options)</code> .
-------------------------	---

	<p> Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Module	N/search Module
Since	Version 2015 Release 2

Values

ACCOUNT	GL_LINES_AUDIT_LOG	RATE_PLAN
ACCOUNTING_BOOK	GOVERNMENT_ISSUED_ID_TYPE	RECENT_RECORD
ACCOUNTING_CONTEXT	HCM_JOB	RESOURCE_ALLOCATION
ACCOUNTING_PERIOD	INTER_COMPANY_JOURNAL_ENTRY	RESTLET
ACTIVITY	INTER_COMPANY_TRANSFER_ORDER	RETURN_AUTHORIZATION
AMORTIZATION_SCHEDULE	INVENTORY_ADJUSTMENT	REVENUE_ARRANGEMENT
AMORTIZATION_TEMPLATE	INVENTORY_COST_REVALUATION	REVENUE_COMMITMENT
ASSEMBLY_BUILD	INVENTORY_COUNT	REVENUE_COMMITMENT_REVERSAL
ASSEMBLY_ITEM	INVENTORY_DETAIL	REVENUE_PLAN
ASSEMBLY_UNBUILD	INVENTORY_ITEM	REV_REC_SCHEDULE
BILLING_ACCOUNT	INVENTORY_NUMBER	REV_REC_TEMPLATE
BILLING_ACCOUNT_BILL_CYCLE	INVENTORY_TRANSFER	ROLE
BILLING_ACCOUNT_BILL_REQUEST	INVOICE	SALES_ORDER
BILLING_CLASS	ISSUE	SALES_TAX_ITEM
BILLING_RATE_CARD	ITEM	SAVED_SEARCH
BILLING_SCHEDULE	ITEM_ACCOUNT_MAPPING	SCHEDULED_SCRIPT_INSTANCE
BIN	ITEM_DEMAND_PLAN	SCRIPT_DEPLOYMENT
BIN_TRANSFER	ITEM_FULFILLMENT	SERIALIZED_ASSEMBLY_ITEM
BIN_WORKSHEET	ITEM_GROUP	SERIALIZED_INVENTORY_ITEM
BLANKET_PURCHASE_ORDER	ITEM_RECEIPT	SERVICE_ITEM
BUNDLE_INSTALLATION_SCRIPT	ITEM_REVISION	SHIP_ITEM
CALENDAR_EVENT	ITEM_SUPPLY_PLAN	SOLUTION
CAMPAIN	JOB	STATISTICAL_JOURNAL_ENTRY
CASH_REFUND	JOB_REQUSITION	SUBSCRIPTION
CASH_SALE	JOURNAL_ENTRY	SUBSCRIPTION_CHANGE_OWNER
CHARGE	KIT_ITEM	SUBSCRIPTION_LINE
CHECK	KUDOS	SUBSCRIPTION_PLAN
CLASSIFICATION	LEAD	
CLIENT_SCRIPT	LOCATION	
COMMERCE_CATEGORY		

■ COMPETITOR	■ LOT_NUMBERED_ASSEMBLY_ITEM	■ SUBSCRIPTION_RENEWAL_HISTORY
■ CONSOLIDATED_EXCHANGE_RATE	■ LOT_NUMBERED_INVENTORY_ITEM	■ SUBSIDIARY
■ CONTACT	■ MANUFACTURING_COST_TEMPLATE	■ SUBTOTAL_ITEM
■ COUPON_CODE	■ MANUFACTURING_OPERATION_TASK	■ SUITELET
■ CREDIT_CARD_CHARGE	■ MANUFACTURING_ROUTING	■ SUITE_SCRIPT_DETAIL
■ CREDIT_CARD_REFUND	■ MAP_REDUCE_SCRIPT	■ SUPPORT_CASE
■ CREDIT_MEMO	■ MARKUP_ITEM	■ TASK
■ CURRENCY	■ MASSUPDATE_SCRIPT	■ TAX_DETAIL
■ CUSTOMER	■ MESSAGE	■ TAX_GROUP
■ CUSTOMER_CATEGORY	■ MFG_PLANNED_TIME	■ TAX_PERIOD
■ CUSTOMER_DEPOSIT	■ NEXUS	■ TAX_TYPE
■ CUSTOMER_PAYMENT	■ NON_INVENTORY_ITEM	■ TERM
■ CUSTOMER_PAYMENT_AUTHORIZATION	■ NOTE	■ TERMINATION_REASON
■ CUSTOMER_REFUND	■ OPPORTUNITY	■ TIME_BILL
■ CUSTOM_TRANSACTION	■ ORGANIZATION_VALUE	■ TIME_OFF_CHANGE
■ DELETED_RECORD	■ OTHER_CHARGE_ITEM	■ TIME_OFF_PLAN
■ DEPARTMENT	■ OTHER_GOVERNMENT_ISSUED_ID	■ TIME_OFF_REQUEST
■ DEPOSIT	■ OTHER_NAME	■ TIME_OFF_RULE
■ DEPOSIT_APPLICATION	■ PARTNER	■ TIME_OFF_TYPE
■ DESCRIPTION_ITEM	■ PASSPORT	■ TOPIC
■ DISCOUNT_ITEM	■ PAYCHECK	■ TRANSACTION
■ DOWNLOAD_ITEM	■ PAYCHECK_JOURNAL	■ TRANSFER_ORDER
■ DRIVERS_LICENSE	■ PAYMENT_ITEM	■ UBER
■ EMPLOYEE	■ PAYROL_BATCH	■ UNITS_TYPE
■ END_TO_END_TIME	■ PAYROLL_ITEM	■ USEVENT_SCRIPT
■ ENTITY	■ PHONE_CALL	■ VENDOR
■ ENTITY_ACCOUNT_MAPPING	■ PORTLET	■ VENDOR_BILL
■ ESTIMATE	■ POSITION	■ VENDOR_CATEGORY
■ EXPENSE_CATEGORY	■ PRICE_LEVEL	■ VENDOR_CREDIT
■ EXPENSE_REPORT	■ PROJECT_EXPENSE_TYPE	■ VENDOR_PAYMENT
■ FAIR_VALUE_PRICE	■ PROJECT_TASK	■ VENDOR_RETURN_AUTHORIZATION
■ FOLDER	■ PROJECT_TEMPLATE	■ WEBSITE
■ GENERIC_RESOURCE	■ PROMOTION_CODE	■ WORKFLOW_ACTION_SCRIPT
■ GIFT_CERTIFICATE	■ PROSPECT	■ WORK_ORDER
■ GIFT_CERTIFICATE_ITEM	■ PURCHASE_CONTRACT	■ WORK_ORDER_CLOSE
■ GLOBAL_ACCOUNT_MAPPING	■ PURCHASE_ORDER	■ WORK_ORDER_COMPLETION
	■ PURCHASE_REQUISITION	■ WORK_ORDER_ISSUE
		■ WORKPLACE

Syntax



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

The sftp module provides a way to upload and download files from external SFTP servers.

SFTP servers can be hosted by your organization or by a third party. NetSuite does not provide SFTP server functionality.

All SFTP transfers to or from NetSuite must originate from SuiteScript. It is not possible for external clients to initiate file transfers using SFTP.



Note: To use an external server to initiate a NetSuite file transfer that doesn't use SFTP, you can use RESTlets or SuiteTalk (Web Services). In SuiteScript, RESTlets can respond to requests containing file data and save them in the File Cabinet. RESTlets can also respond to requests for file data by loading the contents from the File Cabinet and returning them in the response. Note that binary file content must be received or sent as Base64 encoded Strings. See [RESTlet Script Type](#) for more information.

In SuiteTalk, applications can invoke CRUD operations on the File Record to populate or change the contents of the File Cabinet. See the help topics [SuiteTalk \(Web Services\) Platform Guide](#) and [File](#) for more information.

- [N/sftp Module Members](#)
- [Connection Object Members](#)
- [N/sftp Module Script Sample](#)
- [Setting up an SFTP Transfer](#)
- [SFTP Authentication](#)
- [Supported Cipher Suites and Host Key Types](#)
- [Supported SuiteScript File Types](#)

N/sftp Module Members

Member Type	Name	Return Type / Value Type	Description
Object	sftp.Connection	Object	Represents a connection to the account on the remote FTP server.

Member Type	Name	Return Type / Value Type	Description
Method	sftp.createConnection(options)	sftp.Connection	Establishes a connection to a remote FTP server.

Connection Object Members

Member Type	Name	Return Type / Value Type	Description
Method	Connection.download(options)	file.File	Downloads a file from the remote FTP server
	Connection.upload(options)	void	Uploads a file to the remote FTP server.

N/sftp Module Script Sample

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

⚠ Important: Before you run this script, you must replace the GUID and host key with one specific to your account. The user name, URL, and directory values in this sample are also placeholders. Before using this sample, replace the placeholder values with valid values from your NetSuite account.

The following example uploads and downloads a file.

To obtain a real host key, use `ssh-keyscan <domain>`.

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

```
/**
 *@NApiVersion 2.x
 */
require(['N/sftp', 'N/file'],
    function(sftp, file) {
        var myPwdGuid = "B34672495064525E5D65032D63B52301";
        var myHostKey = "AAA1234567890Q=";

        var connection = sftp.createConnection({
            username: 'myuser',
            passwordGuid: myPwdGuid,
            url: 'host.somewhere.com',
            directory: 'myuser/wheres/my/file',
            hostKey: myHostKey
        });

        var myFileToUpload = file.create({
            name: 'originalname.js',
            ...
        });
    }
);
```

```

        fileType: file.fileType.PLAINTEXT,
        contents: 'I am a test file. Hear me roar.'
    });

connection.upload({
    directory: 'relative/path/to/remote/dir',
    filename: 'newFileNameOnServer.js',
    file: myFileToUpload,
    replaceExisting: true
});

var downloadedFile = connection.download({
    directory: 'relative/path/to/file',
    filename: 'downloadMe.js'
});
}
);

```

Setting up an SFTP Transfer

- Development Preparation for SFTP transfers
- Execution of an SFTP transfer

Development Preparation for SFTP transfers

To successfully connect to your SFTP server with SuiteScript, the following steps are recommended:

1. Talk to your SFTP service provider about your plans.
 - Determine the connection properties required to connect with your external SFTP server. For example:
 - username
 - password
 - url
 - port
 - upload/download directories
 - host key
 - host key type
 - Make sure that you know your provider's practices around host key changes, maintenance and failover. For example, find out if there are multiple URLs or ports to try.
 - Check compatibility with NetSuite's supported SFTP ciphers. 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 ad-hoc 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](#)

Only username/password based authentication is supported. Public key based authentication is not supported.

Credential Tokenization

SuiteScript provides the ability for users to securely store authentication credentials in such a way that scripts are able to utilize encrypted saved credentials without being able to see their contents. The script author must specify which scripts and domains are permitted for use with the credential. To restrict the credential for use by SuiteScript automation triggered by the same user who originally saved the credential, the script author can set the `restrictToCurrentUser` parameter.

Creating a Suitelet Form that Contains a Credential Field

```
...
if(request.method === context.Method.GET){
  var form = ui.createForm({title: 'Enter SFTP Credentials'});
  form.addCredentialField({
    id: 'custfield_sftp_password_token',
    label: 'SFTP Password',
  });
}
```

```

        restrictToScriptIds: ['customscript_sftp_script'],
        restrictToDomains: ['acmebank.com'],
        restrictToCurrentUser: true //Depends on use case
    });
    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. SFTP/SSH host keys have no relationship to SFTP key based authentication, which is not currently supported.

By design, there is no SuiteScript API call for checking the host key of a remote SFTP server, or an option to disable strict host key checking. The script must always know the host key ahead of time.

We recommend using OpenSSH's ssh-keyscan tool to check the host key of an external SFTP site. See [Retrieving the Host Key of an External SFTP Server](#).

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+cQx9Jq9UeZja1245X7SBDcRiKh+Sok56VzSw==
```

It is recommended to always pass the key type and port number. This practice helps to avoids 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. Only CTR (and not CBC) ciphers are allowed. Your SFTP server can use the following encryption algorithms:</p> <ul style="list-style-type: none"> ■ AES 128-CTR ■ AES 192-CTR ■ AES 256-CTR <p>Files are not additionally encrypted during transfer. The entire transmission is encrypted by the SSH protocol.</p>
Authentication mechanism	Username/password only (not key-based)
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 <code>Form.addCredentialField(options)</code>. The password guid field's originating credential field must include the SFTP domain on the <code>restrictToDomains</code> parameter. 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 ad-hoc files created using SuiteScript API calls such as `file.create(options)` or `Connection.download(options)`.

File Cabinet and ad hoc files are supported by `Connection.upload(options)`.

Note that `Connection.download(options)` returns an ad-hoc file object. For an ad-hoc 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 an encrypted format in accordance with that provider's security practices. Make sure that you understand your provider's expectations and the cryptographic capabilities in SuiteScript (see [N/crypto Module](#)).

Annotated Syntax Sample

```
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 (defau
       lts to 22).
       */
      port: 22,
      /*
       * The transfer directory supplied by the administrator of the external SFTP server (opt
       ional).
    });
  }
);
```

```

/*
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.fileType.PLAINTEXT,
  contents: 'I am a test file. Hear me roar.'
});

/*
Uploading the file to the external SFTP server.
*/
connection.upload({
  /*
  Subdirectory within the transfer directory specified when connecting (optional).
  */
  directory: 'relative/path/to/remote/dir',
  /*
  Alternate file name to use instead of the one given to the file object (optional).
  */
  filename: 'newFileNameOnServer.js',
  /*
  The file to upload.
  */
  file: myFileToUpload,
  /*
  If a file already exists with that name, replace it instead of failing the upload.
  */
  replaceExisting: true
});

var downloadedFile = connection.download({
  /*
  Subdirectory within the transfer directory specified when connecting (optional).
  */
  directory: 'relative/path/to/file',
  /*
  The name of the file within the above directory on the external SFTP server which to
download.
  */
  filename: 'downloadMe.js'
});

});

```

sftp.Connection

Object Description	Represents a connection to the account on the remote FTP server.
Supported Script Types	Server-side scripts
Module	N/sftp Module
Since	2016.2

Syntax

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

```
//Add additional code ...
var objConnection = sftp.createConnection({
    username: 'username',
    passwordGuid: pwdGuid,
    url: 'host.somewhere.com',
    directory: 'username/wheres/my/file'
    hostKey: myHostKey
});
...
//Add additional code
```

Connection.download(options)

Method Description	Downloads a file from the remote FTP server
Returns	file.File Object
Supported Script Types	Server-side scripts
Governance	100
Module	N/sftp Module
Since	2016.2

Parameters

i 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
			⚠ Important: 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_INVALID_DIRECTORY	The directory does not exist on the remote FTP server.
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_DOES_NOT_EXIST	The options.filename does not exist in the options.directory location.
FTP_PERMISSION_DENIED	Access to the file or directory on the remote FTP server was denied.

Syntax



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

```
//Add additional code
...
var downloadedFile = objConnection.download({
    directory: 'relative/path/to/file',
    filename: 'downloadMe.js'
});
...
//Add additional code
```

Connection.upload(options)

Method Description	Uploads a file to the remote FTP server.
Returns	void

Supported Script Types	Server-side scripts
Governance	100
Module	N/sftp Module
Since	2016.2

Parameters

<p>Note: The options parameter is a JavaScript object.</p>				
Parameter	Type	Required / Optional	Description	Since
options.file	file.File	Required	The file to upload.	2016.2
options.filename	string	Optional	<p>The name to give the uploaded file on server. By default, the filename is the same specified by options.file.</p> <p>Note: Illegal characters are automatically escaped.</p>	2016.2
options.directory	string	Optional	<p>The relative path to the directory where the file should be upload to. By default, the path is set to the current directory.</p> <p>Important: This input must take the form of a relative path.</p>	2016.2
options.timeout	number	Optional	The number of seconds to allow for the file to upload. By default, this value is set to 300 seconds.	2016.2
options.replaceExisting	boolean true false	Optional	Indicates whether the file being uploaded should overwrite any file with the name options.filename that already exists in options.directory. If false, the FTP_FILE_ALREADY_EXISTS exception is thrown when a file with the same name already exists in the options.directory. By default, this value is false.	2016.2

Errors

Error Code	Thrown If
FTP_INVALID_DIRECTORY	The directory does not exist on the remote FTP server.

Error Code	Thrown If
FTP_TRANSFER_TIMEOUT_EXCEEDED	The transfer is taking longer than the specified options.timeout value.
FTP_INVALID_TRANSFER_TIMEOUT	The options.timeout value is either a negative value, zero or greater than 300 seconds.
FTP_FILE_ALREADY_EXISTS	The options.replaceExisting value is false and a file with the same name exists in the remote directory.
FTP_PERMISSION_DENIED	Access to the file or directory on the remote FTP server was denied.

Syntax



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

```
//Add additional code ...
objConnection.upload({
    directory: 'relative/path/to/remote/dir',
    filename: 'newFileNameOnServer',
    file: myFileToUpload,
    replaceExisting: true
});
...
//Add additional code
```

sftp.createConnection(options)

Method Description	Establishes a connection to a remote FTP server. To generate the passwordguid, you can create a suitelet that uses Form.addCredentialField(options) . Use the N/https Module to fetch the GUID value returned from the Suitelet's credential field. For more information, see Setting up an SFTP Transfer
Returns	sftp.Connection , representing that connection.
Supported Script Types	All server-side scripts
Governance	None
Module	N/sftp Module
Since	2016.2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.url	string	Required	The host of the remote account.	2016.2
options.passwordGuid	string	Required	The password GUID for the remote account.	2016.2
options.hostKey	string	Required	The host key for the trusted fingerprint on the server.	2016.2
options.username	string	Optional	The username of the remote account. By default, the login is anonymous.	2016.2
options.port	number	Optional	The port used to connect to the remote account. By default, port 22 is used.	2016.2
options.directory	string	Optional	The remote directory of the connection. Note: The directory property is required if you use a remote server cannot resolve relative paths.	2016.2
options.timeout	number	Optional	The number of seconds to allow for an established connection. By default, this value is set to 20 seconds.	2016.2
options.hostKeyType	string	Optional	The type of host key specified by options.hostKey. This value can be set to one of the following options: <ul style="list-style-type: none">■ dsa■ ecdsa■ rsa	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)

Syntax



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

```
//Add additional code
...
var objConnection = sftp.createConnection({
    username: 'username',
    passwordGuid: pwdGuid,
    url: 'host.somewhere.com',
    directory: 'username/wheres/my/file'
    hostKey: myHostKey
});
...
//Add additional code
```

N/sso Module

Use the sso module to generate outbound single sign-on (SuiteSignOn) tokens. For example, to create a reference to a SuiteSignOn record, or to integrate with an external application.

For more information about NetSuite's SuiteSignOn feature, see the help topic [Outbound Single Sign-on \(SuiteSignOn\)](#).



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

- N/sso Module Member
- N/sso Module Script Sample

N/sso Module Member

Member Type	Name	Return Type	Description
Method	sso.generateSuiteSignOnToken(options)	string	Generates a new SuiteSignOn token for a user

N/sso Module Script Sample

The following sample script shows how you can use the sso module.

These sample scripts use the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see [SuiteScript 2.0 – Script Architecture](#) and [SuiteScript 2.0 Script Types and Entry Points](#).



Important: Before you run this script, you must replace the ID for the SuiteSignOn record with a value specific to your account. Additionally, the SuiteSignOn record you reference must be associated with a specific script. You make this association in the SuiteSignOn record's Connection Points sublist. For help with SuiteSignOn records, see the help topic [Creating SuiteSignOn Records](#).

The following example generates a new OAuth token for a user. The SuiteSignOn feature must be enabled.

```
/** 
 * @NApiVersion 2.x
 */
require(['N/sso'],
    function(sso) {
        function generateSSOToken() {
            var suiteSignOnRecordId = 1;
            var url = sso.generateSuiteSignOnToken(suiteSignOnRecordId);
        }
        generateSSOToken();
});
```

sso.generateSuiteSignOnToken(options)

Method Description	Method used to generate a new SuiteSignOn token for a user.
--------------------	---

	<p>Note: To use this method, Outbound Single Sign-on and web services must be enabled in your account. To enable these features, go to Setup > Company > Enable Features. On the SuiteCloud tab, in the Manage Authentication section, select the SuiteSignOn check box. In the SuiteTalk section, select the Web Services check box. Click Save.</p>
Returns	URL, OAuth token, and any integration variables as a string
Supported Script Types	Portlet scripts, user event scripts, and Suitelets
Governance	20 units
Module	N/sso Module
Since	Version 2015 Release 2

Parameters

<p>Note: The options parameter is a JavaScript object.</p>										
<table border="1"> <thead> <tr> <th>Parameter</th> <th>Type</th> <th>Required / Optional</th> <th>Description</th> <th>Since</th> </tr> </thead> <tbody> <tr> <td>options.suiteSignOnId</td> <td>string</td> <td>required</td> <td> <p>The scriptId specified on the SuiteSignOn record. To see a list of IDs for SuiteSignOn records, go to the SuiteSignOn list page (Setup > Integration > SuiteSignOn).</p> <p>Note: NetSuite recommends that you create a custom scriptId for each SuiteSignOn record to avoid naming conflicts should you decide to use SuiteBundler to deploy your scripts into other accounts.</p> </td> <td>Version 2015 Release 2</td> </tr> </tbody> </table>	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 > Integration > SuiteSignOn).</p> <p>Note: NetSuite recommends that you create a custom scriptId for each SuiteSignOn record to avoid naming conflicts should you decide to use SuiteBundler to deploy your scripts into other accounts.</p>	Version 2015 Release 2
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 > Integration > SuiteSignOn).</p> <p>Note: NetSuite recommends that you create a custom scriptId for each SuiteSignOn record to avoid naming conflicts should you decide to use SuiteBundler to deploy your scripts into other accounts.</p>	Version 2015 Release 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 suiteSignOnId input parameter is invalid or does not exist.

Error Code	Message	Thrown If
		Note: The <code>suiteSignOnId</code> input parameter must be a <code>scriptId</code> and not a internal id.
SSO_CONFIG_REQD	The SuiteSignOn object {1} is not configured for use with this script. You must specify the script as a connection point for this SuiteSignOn.	The <code>suiteSignOnId</code> 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('customssol');
...
//Add additional code
```

N/task Module

Load the task module to create tasks and place them in the internal NetSuite scheduling or task queue. Use the task module to schedule scripts, run Map/Reduce scripts, import CSV files, merge duplicate records, and execute asynchronous workflows.

Each task type has its own corresponding object types. Use the methods available to each object type to configure, submit, and monitor the tasks.

Note: Regardless of task type, tasks are always triggered asynchronously.

- [N/task Module Members](#)
- [ScheduledScriptTask Object Members](#)
- [ScheduledScriptTaskStatus Object Members](#)
- [MapReduceScriptTask Object Members](#)
- [MapReduceScriptTaskStatus Object Members](#)
- [CsvImportTask Object Members](#)
- [CsvImportTaskStatus Object Members](#)
- [EntityDeduplicationTask Object Members](#)
- [EntityDeduplicationTaskStatus Object Members](#)
- [SearchTask Object Members \(BETA\)](#)
- [SearchTaskStatus Object Members \(BETA\)](#)
- [WorkflowTriggerTask Object Members](#)
- [WorkflowTriggerTaskStatus Object Members](#)

- N/task Module Script Sample

N/task Module Members

Member Type	Name	Return Type / Value Type	Description
Object	task.ScheduledScriptTask	Object	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.
	task.ScheduledScriptTaskStatus	Object	Encapsulates the status of a scheduled script placed into the NetSuite scheduling queue.
	task.MapReduceScriptTask	Object	Encapsulates the properties required to run a Map/Reduce script in NetSuite. Use this object to place a Map/Reduce script deployment into the NetSuite task queue.
	task.MapReduceScriptTaskStatus	Object	Encapsulates the properties and status of a Map/Reduce script deployment placed into the NetSuite task queue.
	task.CsvImportTask	Object	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.
	task.CsvImportTaskStatus	Object	Encapsulates the status of a CSV import task placed into the NetSuite scheduling queue.
	task.EntityDeduplicationTask	Object	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	Encapsulates the status of a merge duplicate record task placed into the NetSuite task queue.
	task.SearchTask	Object	Encapsulates the properties required to initiate an asynchronous search.
	task.SearchTaskStatus	Object	Encapsulates the status of an asynchronous search initiation task that is placed into the NetSuite task queue.
	task.WorkflowTriggerTask	Object	Encapsulates all the properties required to

Member Type	Name	Return Type / Value Type	Description
			asynchronously initiate a workflow. Use WorkflowTriggerTask to create a task that initiates an instance of a specific workflow.
	task.WorkflowTriggerTaskStatus	Object	Encapsulates the status of an asynchronous workflow initiation task placed into the NetSuite task queue.
Method	task.create(options)	task.ScheduledScriptTask task.MapReduceScriptTask task.CsvImportTask task.EntityDeduplicationTask task.SearchTask task.WorkflowTriggerTask	Creates an object for a specific task type and returns the task object.
	task.checkStatus(options)	task.ScheduledScriptTaskStatus task.MapReduceScriptTaskStatus task.CsvImportTaskStatus task.EntityDeduplicationTaskStatus task.SearchTaskStatus task.WorkflowTriggerTaskStatus	Returns a task status object associated with a specific task ID.
Enum	task.TaskType	enum	Enumeration that holds the string values for the types of task objects, supported by the N/task Module , that you can create with <code>task.create(options)</code> .
	task.TaskStatus	enum	Enumeration that holds the string values for the possible status of tasks created and submitted with the N/task Module .
	task.MasterSelectionMode	enum	Enumeration that holds the string values for supported master selection modes when merging duplicate records with <code>task.EntityDeduplicationTask</code> .
	task.DedupeMode	enum	Enumeration that holds the string values for available deduplication modes when merging duplicate records with <code>task.EntityDeduplicationTask</code> .
	task.DedupeEntityType	enum	Enumeration that holds the string values for entity types for which you can merge duplicate records with <code>task.EntityDeduplicationTask</code> .
	task.MapReduceStage	enum	Enumeration that holds the string values for possible stages in <code>task.MapReduceScriptTask</code> for a map/reduce script.

ScheduledScriptTask Object Members

The following members are called on `task.ScheduledScriptTask`.

Member Type	Name	Return Type / Value Type	Description
Method	ScheduledScriptTask.submit()	string	Directs NetSuite to place a scheduled script deployment into the NetSuite scheduling queue and returns a unique ID for the task.
Property	ScheduledScriptTask.scriptId	number string	Internal ID (as a number), or script ID (as a string) for the script record associated with a task.ScheduledScriptTask object.
	ScheduledScriptTask.deploymentId	number string	Internal ID (as a number), or script ID (as a string), for the script deployment record associated with a task.ScheduledScriptTask object.
	ScheduledScriptTask.params	Object	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	Description
Property	ScheduledScriptTaskStatus.scriptId	read-only number	Internal ID for a script record associated with a specific task.ScheduledScriptTask object.
	ScheduledScriptTaskStatus.deploymentId	read-only number	Internal ID for a script deployment record associated with a specific task.ScheduledScriptTask object.
	ScheduledScriptTaskStatus.status	task.TaskStatus	Status for a scheduled script task. Returns a task.TaskStatus enum value.

MapReduceScriptTask Object Members

The following members are called on task.MapReduceScriptTask.

Member Type	Name	Return Type / Value Type	Description
Method	MapReduceScriptTask.submit()	string	Directs NetSuite to place a map/reduce script deployment into the NetSuite task queue and returns a unique ID for the task.
Property	MapReduceScriptTask.scriptId	number string	Internal ID (as a number), or script ID (as a string), for the map/reduce script record.

Member Type	Name	Return Type / Value Type	Description
	MapReduceScriptTask.deploymentId	number string	Internal ID (as a number), or script ID (as a string), for the script deployment record for a map/reduce script.
	MapReduceScriptTask.params	Object	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	Description
Method	MapReduceScriptTaskStatus.getPercentageCompleted()	number	Returns the current percentage complete for the current stage of a task.MapReduceScriptTask .
	MapReduceScriptTaskStatus.getPendingMapCount()	number	Returns the total number of records or rows not yet processed by the map stage of a task.MapReduceScriptTask .
	MapReduceScriptTaskStatus.getTotalMapCount()	number	Returns the total number of records or rows passed as input to the map stage of a task.MapReduceScriptTask .
	MapReduceScriptTaskStatus.getPendingMapSize()	number	Returns the total number of bytes not yet processed by the map stage, as a component of total size, of a task.MapReduceScriptTask .
	MapReduceScriptTaskStatus.getPendingReduceCount()	number	Returns the total number of records or rows not yet processed by the reduce stage of a task.MapReduceScriptTask .
	MapReduceScriptTaskStatus.getTotalReduceCount()	number	Returns the total number of record or row inputs to the reduce stage of a task.MapReduceScriptTask .
	MapReduceScriptTaskStatus.getPendingReduceSize()	number	Returns the total number of bytes not yet processed by the reduce stage, as a component of total size, of a task.MapReduceScriptTask .

Member Type	Name	Return Type / Value Type	Description
	MapReduceScriptTaskStatus.getPendingOutputCount()	number	Returns the total number of records or rows not yet processed by a task.MapReduceScriptTask .
	MapReduceScriptTaskStatus.getPendingOutputSize()	number	Returns the total size in bytes of all key/value pairs written as output, as a component of total size, by a task.MapReduceScriptTask .
	MapReduceScriptTaskStatus.getTotalOutputCount()	number	Returns the total number of records or rows passed as inputs to the OUTPUT phase of a task.MapReduceScriptTask .
	MapReduceScriptTaskStatus.getCurrentTotalSize()	number	Returns the total size in bytes of all stored work in progress by a task.MapReduceScriptTask .
Property	MapReduceScriptTaskStatus.scriptId	read-only number string	Internal ID for a map/reduce script record associated with a specific task.MapReduceScriptTask .
	MapReduceScriptTaskStatus.deploymentId	read-only number string	Internal ID for a script deployment record associated with a specific task.MapReduceScriptTask .
	MapReduceScriptTaskStatus.status	task.TaskStatus	Status for a map/reduce script task. Returns a task.TaskStatus enum value.
	MapReduceScriptTaskStatus.stage	task.MapReduceStage	Current stage of processing for a Map/Reduce script task. See task.MapReduceStage for supported values.

CsvImportTask Object Members

The following members are called on [task.CsvImportTask](#).

Member Type	Name	Return Type / Value Type	Description
Method	CsvImportTask.submit()	string	Directs NetSuite to place a CSV import task into the NetSuite task queue and returns a unique ID for the task.
Property	CsvImportTask.importFile	file.File string	CSV file to import. Use a file.File object or a string that represents the CSV text to be imported.

Member Type	Name	Return Type / Value Type	Description
	CsvImportTask.mappingId	number string	Script ID or internal ID of the saved import map that you created when you ran the Import Assistant.
	CsvImportTask.queueId	number	Overrides the Queue Number property under Advanced Options on the Import Options page of the Import Assistant.
	CsvImportTask.name	string	Name for the CSV import task.
	CsvImportTask.linkedFiles	Object	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	Description
Property	CsvImportTaskStatus.status	task.TaskStatus	Status for a CSV import task. Returns a task.TaskStatus enum value.

EntityDeduplicationTask Object Members

The following members are called on [task.EntityDeduplicationTask](#).

Member Type	Name	Return Type / Value Type	Description
Method	EntityDeduplicationTask.submit()	string	Directs NetSuite to place the merge duplicate records task into the NetSuite task queue and returns a unique ID for the task.
Property	EntityDeduplicationTask.entityType	task.DedupeEntityType	Sets the type of entity on which you want to merge duplicate records.
	EntityDeduplicationTask.masterRecordId	number	When you merge duplicate records, you can delete all duplicates for a record or merge information from the duplicate records into the master record.

Member Type	Name	Return Type / Value Type	Description
	EntityDeduplicationTask.masterSelectionMode	task.MasterSelectionMode	When you merge duplicate records, you can delete all duplicates for a record or merge information from the duplicate records into the master record.
	EntityDeduplicationTask.dedupeMode	task.DedupeMode	Sets the mode in which to merge or delete duplicate records.
	EntityDeduplicationTask.recordIds	number[]	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	Description
Property	EntityDeduplicationTaskStatus.status	task.TaskStatus	Status for a merge duplicate record task.

SearchTask Object Members (BETA)



Important: This Object is part of a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any changes may impact the feature's operation with the NetSuite application. NetSuite warranties and product service levels shall not apply to the feature or the impact of the feature on other portions of the NetSuite application. NetSuite may review and monitor the performance and use of the feature.

The following members are called on [task.SearchTask](#).

Member Type	Name	Return Type / Value Type	Description
Method	SearchTask.submit()	string	Used to place the asynchronous search initiation task into the SuiteScript task queue, and return a unique ID for the task.
Property	SearchTask.fileId	number	ID of the saved search to be executed during the task.
	SearchTask.filePath	string	ID of the CSV file to export search results into.
	SearchTask.savedSearchID	number	Path of the CSV file to export search results into.

SearchTaskStatus Object Members (BETA)



Important: This Object is part of a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any changes may impact the feature's operation with the NetSuite application. NetSuite warranties and product service levels shall not apply to the feature or the impact of the feature on other portions of the NetSuite application. NetSuite may review and monitor the performance and use of the feature.

The following members are called on [task.SearchTaskStatus](#).

Member Type	Name	Return Type / Value Type	Description
Property	SearchTaskStatus.fileId	number	ID of the CSV file into which search results are exported.
	SearchTaskStatus.savedSearchId	number	ID of the saved search executed during the task.
	SearchTaskStatus.status	task.TaskStatus	Status of an asynchronous search task placed in the NetSuite task queue.
	SearchTaskStatus.taskId	number	ID of the asynchronous task.

WorkflowTriggerTask Object Members

The following members are called on [task.WorkflowTriggerTask](#).

Member Type	Name	Return Type / Value Type	Description
Method	WorkflowTriggerTask.submit()	string	Directs NetSuite to place the asynchronous workflow initiation task into the NetSuite scheduling queue and returns a unique ID for the task.
Property	WorkflowTriggerTask.recordType	string	Record type of the workflow base record. For example, customer, salesorder, or lead.
	WorkflowTriggerTask.recordId	number	Internal ID of the workflow definition base record. For example, 55 or 124.
	WorkflowTriggerTask.workflowId	number string	Internal ID (as a number), or script ID (as a string), for the workflow definition.
	WorkflowTriggerTask.params	Object	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	Description
Property	WorkflowTriggerTaskStatus .status	task.TaskStatus	Status for a asynchronous workflow placed in the NetSuite task queue.

N/task Module Script Sample

Sample 1

The following example creates and submits a map reduce script job and sends an email on failure.

```
/**
 * @NApiVersion 2.x
 */
// This example shows how to create and submit a map reduce script job and sends an email on failure
require(['N/task', 'N/runtime', 'N/email'],
    function(task, runtime, email) {
        function createAndSubmitMapReduceJob() {
            var mapReduceScriptId = 34;
            log.audit('mapreduce id: ', mapReduceScriptId);
            var mrTask = task.create({
                taskType: task.TaskType.MAP_REDUCE,
               scriptId: mapReduceScriptId,
                deploymentId : 1
            });
            var mrTaskId = mrTask.submit();
            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.'
                });
            }
        }
        createAndSubmitMapReduceJob();
    });
}
```

Sample 2



Important: The asynchronous search task is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any changes may impact the feature's operation with the NetSuite application. NetSuite warranties and product service levels shall not apply to the feature or the impact of the feature on other portions of the NetSuite application. NetSuite may review and monitor the performance and use of the feature.

The following example creates an asynchronous search task intended to execute a search and export results of the saved search into a CSV file stored in the file cabinet. After the task is submitted, the script needs the task ID to retrieve the task status.



Note: Note: Some of the values in this sample are placeholders. Before using this sample, 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.

```
require(['N/task'],
  function(task)
{
  // obtain saved search id - for example, create temporary search based on the following
  // snippet
  // var mySearch = search.create({
  //   type: search.Type.SALES_ORDER,
  //   id: 'customsearch_my_search',
  //   filters: [...],
  //   columns: [...]});
  // mySearch.save();
  // var savedSearchId = mySearch.searchId;
  var savedSearchId = -10;

  //creating async search task
  var myTask = task.create({
    taskType: task.TaskType.SEARCH
  });
  myTask.savedSearchId = savedSearchId;
  // obtain id of file into which search results are exported - for example, create a fil
  e with a folder of your choice in the file cabinet
  myTask.fileId =448;
  var myTaskId = myTask.submit();
  // store myTaskId for later usage - for example into custom record

  var taskStatus = task.checkStatus({
    taskId: myTaskId
  });
  var fileId = taskStatus.fileId;
  // file id of file being exported to
  var savedSearchId = taskStatus.savedSearchId;
  // id of search being executed
  if(taskStatus.status === task.TaskStatus.COMPLETE) {
    // Add code to handle task complete
    // for example, delete temporary search and delete custom record containing retriev
    ed myTaskId
  }
});
```

task.ScheduledScriptTask

Object Description	Encapsulates all the properties of scheduled script task in SuiteScript. Use this object to place a scheduled script deployment into the NetSuite scheduling queue. To use the <code>ScheduledScriptTask</code> Object:
	<ol style="list-style-type: none"> 1. In the NetSuite UI, create the script record and script deployment record.

-
2. Use `task.create(options)` to create the `ScheduledScriptTask` object.
 3. Use the `ScheduledScriptTask` object properties to set the script and deployment properties.
 4. Use `ScheduledScriptTask.submit()` to deploy the scheduled script to the NetSuite scheduling queue.
 5. Use the properties for the `task.ScheduledScriptTaskStatus` object to get the status of the scheduled script.

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

For more information about scheduled scripts in NetSuite, see [Scheduled Script Type](#).

Supported Script Types	Server-side scripts
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

Method Description	<p>Directs NetSuite to place a scheduled script deployment into the NetSuite scheduling queue and returns a unique ID for the task. Scheduled scripts must meet the following requirements:</p> <ul style="list-style-type: none"> ■ The scheduled script must have a status of Not Scheduled on the Script Deployment page. If the script status is set to Testing on the Script Deployment page, this method will not place the script into the scheduling queue. ■ If the deployment status on the Script Deployment page is set to Scheduled, the script will be placed into the queue according to the time(s) specified on the Script Deployment page. ■ Only administrators can run scheduled scripts. If a user event script calls <code>ScheduledScriptTask.submit()</code>, the user event script has to be deployed with admin permissions.
---------------------------	--

Returns	the task id as a string
Supported Script Types	Server-side scripts
Governance	20 units
Module	N/task Module
Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
var scheduledScriptTaskId = scriptTask.submit();
...
//Add additional code
```

ScheduledScriptTask.scriptId

Property Description	Internal ID (as a number), or script ID (as a string), for the script record associated with a task.ScheduledScriptTask object.
Type	number string
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
var scheduledscriptId = 34;
...
//Add additional code
```

ScheduledScriptTask.deploymentId

Property Description	Internal ID (as a number), or script ID (as a string), for the script deployment record associated with a task.ScheduledScriptTask Object.
Type	number string
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code .
..
scheduledTask.deploymentId = 1;
...
//Add additional code
```

ScheduledScriptTask.params

Property Description	Object with key/value pairs that override static script parameter field values on the script deployment. Use these parameters for the task.ScheduledScriptTask object to programmatically pass values to the script deployment. For more information about script parameters, see the help topic Creating Script Parameters Overview .
Type	object
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
scriptTask.params = {searchId: 'custsearch_456'};
...
//Add additional code
```

task.ScheduledScriptTaskStatus

Object Description	Encapsulates the properties and status of a scheduled script placed into the NetSuite scheduling queue.
---------------------------	---

Use `task.checkStatus(options)` with the unique ID for the scheduled script task to get the `ScheduledScriptTaskStatus` Object. For a complete list of this object's properties, see [ScheduledScriptTaskStatus Object Members](#).

Supported Script Types	Server-side scripts
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
var res = task.checkStatus(scriptTaskId);
log.debug('Initial status: ' + res.status);
...
//Add additional code
```

ScheduledScriptTaskStatus.scriptId

Property Description	Internal ID for a script record associated with a specific <code>task.ScheduledScriptTask</code> Object. Use this ID to get more details about the script record for the scheduled task.
Type	read-only number
Module	N/task Module
Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
log.audit('Initial status: ' + status.scriptId);
...
```

```
//Add additional code
```

ScheduledScriptTaskStatus.deploymentId

Property Description	Internal ID for a script deployment record associated with a specific task.ScheduledScriptTask Object. Use this ID to get more details about the script deployment record for the scheduled task.
Type	number
Module	N/task Module
Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
log.audit({
    details:'Deployment ID: ' + status.scriptId
});
...
//Add additional code
```

ScheduledScriptTaskStatus.status

Property Description	Status for a scheduled script task. Returns a task.TaskStatus enum value.
Type	task.TaskStatus
Module	N/task Module
Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
log.audit({
    details: 'Status: ' + summary.status
});
...
//Add additional code
```

task.MapReduceScriptTask

Object Description	Encapsulates the properties required to run a map/reduce script in NetSuite. Use this object to place a map/reduce script deployment into the NetSuite task queue. To use the <code>MapReduceScriptTask</code> object: <ul style="list-style-type: none"> ■ In the NetSuite UI, create the script record and script deployment records. ■ Use <code>task.create(options)</code> to create the <code>MapReduceScriptTask</code> object. ■ Use the <code>MapReduceScriptTask</code> object properties to set the script and deployment properties. ■ Use <code>MapReduceScriptTask.submit()</code> to deploy the script to the NetSuite task queue. ■ Use the properties for the <code>task.MapReduceScriptTaskStatus</code> object to get the status of the map/reduce script. For a complete list of this object's methods and properties, see MapReduceScriptTask Object Members . For more information about scheduled scripts in NetSuite, see Map/Reduce Script Type .
Supported Script Types	Server-side scripts
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
var mrTask = task.create({taskType: task.TaskType.MAP_REDUCE});
mrTask.scriptId = mapReducescriptId;
mrTask.deploymentId = 1;
var mrTaskId = mrTask.submit();
...
```

```
//Add additional code
```

MapReduceScriptTask.submit()

Method Description	Directs NetSuite to place a map/reduce script deployment into the NetSuite task queue and returns a unique ID for the task. For more information, see task.MapReduceScriptTask .
Returns	string
Supported Script Types	Server-side scripts
Governance	20 units
Module	N/task Module
Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
var mrTaskId = mrTask.submit();
...
//Add additional code
```

MapReduceScriptTask.scriptId

Property Description	Internal ID (as a number), or script ID (as a string), for the map/reduce script record.
Type	number string
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
```

```
...
var mapReducescriptId = 34;
...
//Add additional code
```

MapReduceScriptTask.deploymentId

Property Description	Internal ID (as a number) or script ID (as a string), for the script deployment record for a map/reduce script.
Type	number string
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
mrTask.deploymentId = 1;
...
//Add additional code
```

MapReduceScriptTask.params

Property Description	Object that represents key/value pairs that override static script parameter field values on the script deployment record. Use these parameters on a <code>task.MapReduceScriptTask</code> object to programmatically pass values to the script deployment. For more information about script parameters, see the help topic Creating Script Parameters Overview .
Type	object
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

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

```
mrTask.params = {doSomething: true};
...
//Add additional code
```

task.MapReduceScriptTaskStatus

Object Description	Encapsulates the properties and status of a map/reduce script deployment placed into the NetSuite task queue. Use task.checkStatus(options) with the unique ID for the map/reduce script task to get the <code>MapReduceScriptTaskStatus</code> object. For a complete list of this object's methods and properties, see MapReduceScriptTaskStatus Object Members .
Supported Script Types	Server-side scripts
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
if (summary.stage === task.MapReduceStage.SUMMARIZE)
    log.audit('Almost done...');
...
//Add additional code
```

MapReduceScriptTaskStatus.getPercentageCompleted()

Method Description	Returns the current percentage complete for the current stage of a <code>task.MapReduceScriptTask</code> . Use the <code>MapReduceScriptTaskStatus.stage</code> property to get the current stage.
Note:	The input and summarize stages are either 0% or 100% complete at any time.
Returns	number
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
var completion = taskStatus.getPercentageCompleted();
log.audit('Percentage Completed: ' + completion);
...
//Add additional code
```

MapReduceScriptTaskStatus.getPendingMapCount()

Method Description	Returns the total number of records or rows not yet processed by the map stage of a task.MapReduceScriptTask. Use the MapReduceScriptTaskStatus.stage property to get the current stage.
Returns	number
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
var summary = taskStatus.getPendingMapCount();
log.audit('Pending Map Count: ' + summary);
...
//Add additional code
```

MapReduceScriptTaskStatus.getTotalMapCount()

Method Description	Returns the total number of records or rows passed as input to the map stage of a task.MapReduceScriptTask. Use the MapReduceScriptTaskStatus.stage property to get the current stage.
Returns	number
Supported Script Types	Server-side scripts
Governance	10 units

Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var summary = taskStatus.getTotalMapCount();
log.audit('Total Map Count: ' + summary);
...
//Add additional code
```

MapReduceScriptTaskStatus.getPendingMapSize()

Method Description	Returns the total number of bytes not yet processed by the map stage, as a component of total size, of a task.MapReduceScriptTask . Use the MapReduceScriptTaskStatus.stage property to get the current stage.
Returns	number
Supported Script Types	Server-side scripts
Governance	25 units
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var summary = taskStatus.getPendingMapSize();
log.audit('Pending Map Size: ' + summary);
...
//Add additional code
```

MapReduceScriptTaskStatus.getPendingReduceCount()

Method Description	Returns the total number of records or rows not yet processed by the reduce stage of a task.MapReduceScriptTask . Use the MapReduceScriptTaskStatus.stage property to get the current stage.
---------------------------	--

Returns	number
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var summary = taskStatus.getPendingReduceCount();
log.audit({
    details: 'Pending Reduce Count: ' + summary
});
...
//Add additional code
```

MapReduceScriptTaskStatus.getTotalReduceCount()

Method Description	Returns the total number of record or row inputs to the REDUCE phase of a task.MapReduceScriptTask . Use the MapReduceScriptTaskStatus.stage property to get the current stage.
Returns	number
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var summary = taskStatus.getTotalReduceCount();
log.audit({
    details: 'Reduce Count: ' + summary
});
...
//Add additional code
```

MapReduceScriptTaskStatus.getPendingReduceSize()

Method Description	Returns the total number of bytes not yet processed by the reduce stage, as a component of total size, of a task.MapReduceScriptTask . Use the MapReduceScriptTaskStatus.stage property to get the current stage.
Returns	number
Supported Script Types	Server-side scripts
Governance	25 units
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var summary = taskStatus.getPendingReduceSize();
log.audit({
    details: 'Pending Reduce Size: ' + summary
});
...
//Add additional code
```

MapReduceScriptTaskStatus.getPendingOutputCount()

Method Description	Returns the total number of records or rows not yet processed by a task.MapReduceScriptTask .
Returns	number
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
var total = summary.getPendingOutputCount()
```

```

log.audit({
    title: 'Count',
    details: total
});
...
//Add additional code

```

MapReduceScriptTaskStatus.getPendingOutputSize()

Method Description	Returns the total size in bytes of all key/value pairs written as output, as a component of total size, by a task.MapReduceScriptTask .
Returns	number
Supported Script Types	Server-side scripts
Governance	25 units
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

```

//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
var total = summary.getPendingOutputSize()
log.audit({
    title: 'Size',
    details: total
});
...
//Add additional code

```

MapReduceScriptTaskStatus.getTotalOutputCount()

Method Description	Returns the total number of records or rows passed as inputs to the output phase of a task.MapReduceScriptTask . Use the MapReduceScriptTaskStatus.stage property to get the current stage.
Returns	number
Supported Script Types	Server-side scripts
Governance	10 units
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

MapReduceScriptTaskStatus.getCurrentTotalSize()

Method Description	Returns the total size in bytes of all stored work in progress by a task.MapReduceScriptTask .
Returns	number
Supported Script Types	Server-side scripts
Governance	25 units
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

MapReduceScriptTaskStatus.scriptId

Property Description	Internal ID for a map/reduce script record associated with a specific task.MapReduceScriptTask .
-----------------------------	--

Type	read-only number
Module	N/task Module
Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
log.audit({
    title: 'Script ID',
    details: summary.scriptId
});

...
//Add additional code
```

MapReduceScriptTaskStatus.deploymentId

Property Description	Internal ID for a script deployment record associated with a specific task.MapReduceScriptTask.
Type	read-only number
Module	N/task Module
Since	Version 2015 Release 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 [N/task Module Script Sample](#).

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

```

var summary = task.checkStatus(scriptTaskId);
log.audit({
    title: 'Deployment ID',
    details: summary.deploymentId
});

...
//Add additional code

```

MapReduceScriptTaskStatus.status

Property Description	Status for a Map/Reduce script task. Returns a task.TaskStatus enum value.
Type	task.TaskStatus
Module	N/task Module
Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```

//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
log.audit({
    title: 'Status',
    details: summary.status
});

...
//Add additional code

```

MapReduceScriptTaskStatus.stage

Property Description	Current stage of processing for a Map/Reduce script task. See task.MapReduceStage for supported values.
Type	task.MapReduceStage
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
if (summary.stage === task.MapReduceStage.SUMMARIZE)
    log.audit({
        details: 'Almost done...'
    });
...
//Add additional code
```

task.CsvImportTask

Object Description

Encapsulates the properties of a CSV import task. Use the methods and properties for this object to submit a CSV import task into the task queue and asynchronously import record data into NetSuite.

Use the `CsvImportTask` Object to perform the following types of tasks:

- Automate standard record data import for SuiteApp installations, demo environments, and testing environments.
- Import data on a schedule using a scheduled script.
- Build integrated CSV imports with RESTlets.

Use the following process to import CSV data with `CsvImportTask`:

- In the NetSuite UI, run the Import Assistant to set up the CSV mapping and import options. You must run the Import Assistant to set up the necessary mapping for the CSV import. You can use a sample file or files to set up the mapping. Note the following information:

- Script ID for import map.
- Any required linked files.

For more information, see the help topic [Importing CSV Files with the Import Assistant](#).

- Use `task.create(options)` to create the `CsvImportTask` object.
- Use the `CsvImportTask` object properties to set the script and deployment properties.
- Use `CsvImportTask.submit()` to submit the import task to the NetSuite task queue.
- Use the properties for the `task.CsvImportTaskStatus` object to get the status of the import process.

Use the following guidelines with the `CsvImportTask` Object:

- CSV imports performed within scripts are subject to the existing application limit of 25,000 records.
- You cannot import data that is imported by (2-step) assistants in the UI, because these import types do not support saved import maps. This

limitation applies to budget, single journal entry, single inventory worksheet, project tasks, and Web site redirects imports.

- This object has access only to the field mappings of a saved import map; it does not have access to advanced import options defined in the Import Assistant, such as multi-threading and multiple queues.

Even if you set options to use multiple threads or queues for an import job and then save the import map, these settings are not available to `CsvImportTask`. When this object submits a CSV import job based on the saved import map, a single thread and single queue are used.

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

Supported Script Types	Server-side scripts
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

Method Description	Directs NetSuite to place a CSV import task into the NetSuite task queue and returns a unique ID for the task. Use <code>CsvImportTaskStatus.status</code> to view the status of a submitted task. This method throws errors resulting from inline validation of CSV file data before the import of data begins (the same validation that is performed between the mapping step and the save step in the Import Assistant). Any errors that occur during the import job are recorded in the CSV response file, as they are for imports initiated through the Import Assistant.
Returns	string
Supported Script Types	Server-side scripts
Governance	100 units
Module	N/task Module

Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
var csvImportTaskId = csvTask.submit();
...
//Add additional code
```

CsvImportTask.importFile

Property Description	CSV file to import. Use a file.File object or a string that represents the CSV text to be imported.
Type	file.File string
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var f = file.load('SuiteScripts/custjoblist.csv');
scriptTask.importFile = f;
...
//Add additional code
```

CsvImportTask.mappingId

Property Description	Script ID or internal ID of the saved import map that you created when you ran the Import Assistant. See task.CsvImportTask .
Type	number string

Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
scriptTask.mappingId = 51;
...
//Add additional code
```

CsvImportTask.queueId

Property Description	Overrides the Queue Number property under Advanced Options on the Import Options page of the Import Assistant. Use this property to programmatically select an import queue and improve performance during the import.
Type	number
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
scriptTask.queueId = 2;
...
//Add additional code
```

CsvImportTask.name

Property Description	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.
----------------------	--

Type	string
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
csvTask.name = 'Import Entities'
...
//Add additional code
```

CsvImportTask.linkedFiles

Property Description	A map of key/value pairs that sets the data to be imported in a linked file for a multi-file import job, by referencing a file in the file cabinet or the raw CSV data to import. The key is the internal ID of the record sublist for which data is being imported and the value is either a file.File object or the raw CSV data to import. You can assign multiple types of values to the <code>linkedFiles</code> property.
Type	Object
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
scriptTask.linkedFiles = {'addressbook': 'street,city\nval1,val2', 'purchases': file.load('SuiteScripts/other.csv')};
...
//Add additional code
```

task.CsvImportTaskStatus

Object Description	Encapsulates the status of a CSV import task placed into the NetSuite scheduling queue.
--------------------	---

Use `task.checkStatus(options)` with the unique ID for the CSV import task to get the `CsvImportTaskStatus` object. For a complete list of this object's properties, see [CsvImportTaskStatus Object Members](#).

Supported Script Types	Server-side scripts
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var csvTaskStatus = task.checkStatus({
    taskId: csvTaskId
});
if (csvTaskStatus.status === task.TaskStatus.FAILED)
...
//Add additional code
```

CsvImportTaskStatus.status

Property Description	Status for a CSV import task. Returns a <code>task.TaskStatus</code> enum value.
Type	<code>task.TaskStatus</code>
Module	N/task Module
Since	Version 2015 Release 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 [N/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

Object Description	<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 NetSuite's predefined duplicate detection rules, or you can define your own. After the records are merged or deleted, in the UI, the records no longer appear as duplicates at Lists > Mass Update > Mass Duplicate Record Merge.</p> <p>For more information about merging duplicate records in NetSuite, see the help topic Merging or Deleting Duplicate Records.</p> <p>To use the <code>EntityDeduplicationTask</code> Object:</p> <ul style="list-style-type: none"> ▪ Use <code>task.create(options)</code> to create the <code>EntityDeduplicationTask</code> object. ▪ Use <code>EntityDeduplicationTask.entityType</code> to select the entity type on which you want to merge duplicate records. ▪ Use <code>EntityDeduplicationTask.dedupeMode</code> to select the action to take for the duplicate records. ▪ Use a <code>EntityDeduplicationTask.masterSelectionMode</code> enum value to identify which record to use as the master record in the merge. ▪ If you use <code>MasterSelectionMode.SELECT_BY_ID</code> for the master selection mode, set the ID of the master record with <code>EntityDeduplicationTask.masterRecordId</code>. ▪ Identify the duplicate records. Use the <code>search.duplicates(options)</code> method in the N/search Module to find the duplicate records. ▪ Use <code>EntityDeduplicationTask.submit()</code> to submit the merge duplicate record task to the NetSuite task queue. ▪ Use the properties for the <code>task.EntityDeduplicationTaskStatus</code> object to get the status of the merge duplicate record task. <p>Use the following guidelines with the <code>EntityDeduplicationTask</code> Object:</p> <ul style="list-style-type: none"> ▪ You can only submit 200 records in a single merge duplicate records task. ▪ The merge duplicate functionality on non-entity records is not supported in SuiteScript. ▪ You must have full access to the Duplicate Record Management permission to merge duplicates. <p>For a complete list of this object's methods and properties, see EntityDeduplicationTask Object Members.</p>
Supported Script Types	Server-side scripts
Module	N/task Module

Since	Version 2015 Release 2
-------	------------------------

Syntax



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

Method Description	Directs NetSuite to place the merge duplicate records task into the NetSuite task queue and returns a unique ID for the task. Use EntityDeduplicationTaskStatus.status to view the status of a submitted task.
Returns	task id as a string
Supported Script Types	Server-side scripts
Governance	100 units
Module	N/task Module
Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
var dedupeTaskId = dedupeTask.submit();
...
```

```
//Add additional code
```

EntityDeduplicationTask.entityType

Property Description	Sets the type of entity on which you want to merge duplicate records. Use a task.DedupeEntityType enum value to set the value.
	Note: If you set entityType to CUSTOMER, the system will automatically include prospects and leads in the task request.
Type	<code>task.DedupeEntityType</code>
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
...
dedupeTask.entityType = task.DedupeEntityType.CUSTOMER;
...
//Add additional code
```

EntityDeduplicationTask.masterRecordId

Property Description	When you merge duplicate records, you can delete all duplicates for a record or merge information from the duplicate records into the master record. Use this property to set the ID of the master record that you want to use as the master record in the merge.
	Important: You must also select SELECT_BY_ID for the EntityDeduplicationTask.masterSelectionMode property, or NetSuite ignores this setting.
Type	<code>number</code>
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
```

```

...
dedupeTask.masterSelectionMode = task.MasterSelectionMode.SELECT_BY_ID;
dedupeTask.masterRecordId = 107;
...
//Add additional code

```

EntityDeduplicationTask.masterSelectionMode

Property Description	When you merge duplicate records, you can delete all duplicates for a record or merge information from the duplicate records into the master record. Set this property to determine which of the duplicate records to keep or select the master record to use by ID. Use EntityDeduplicationTask.masterSelectionMode to set the value.
Type	task.MasterSelectionMode
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```

//Add additional code
...
dedupeTask.masterSelectionMode = task.MasterSelectionMode.MOST_RECENT_ACTIVITY;
...
//Add additional code

```

EntityDeduplicationTask.dedupeMode

Property Description	Sets the mode in which to merge or delete duplicate records. Use a EntityDeduplicationTask.dedupeMode enum value to set the value.
Type	task.DedupeMode
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```

//Add additional code
...
dedupeTask.dedupeMode = task.DedupeMode.MERGE;
...

```

```
//Add additional code
```

EntityDeduplicationTask.recordIds

Property Description	Number array of record internal IDs to perform the merge or delete operation on. You can use the search.duplicates(options) method to identify duplicate records or create an array with record internal IDs.
Type	number[]
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
dedupeTask.recordIds = ['107', '110'];
...
//Add additional code
```

task.EntityDeduplicationTaskStatus

Object Description	Encapsulates the status of a merge duplicate record task placed into the NetSuite task queue by EntityDeduplicationTask.submit() . Use task.checkStatus(options) with the unique ID for the merge duplicate records task to get this Object. For a complete list of this object's properties, see EntityDeduplicationTaskStatus Object Members .
Supported Script Types	Server-side scripts
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var dedupeTaskStatus = task.checkStatus({
    taskId: taskId
});
if (dedupeTaskStatus.status === task.TaskStatus.FAILED)
```

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

EntityDeduplicationTaskStatus.status

Property Description	Status for a merge duplicate record task. Returns a task.TaskStatus enum value.
Type	task.TaskStatus
Module	N/task Module
Since	Version 2015 Release 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 [N/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

 **Important:** This is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any changes may impact the feature's operation with the NetSuite application. NetSuite warranties and product service levels shall not apply to the feature or the impact of the feature on other portions of the NetSuite application. NetSuite may review and monitor the performance and use of the feature.

Object Description	Encapsulates the properties of a search task. Use the methods and properties for this object to submit a search task into the task queue, execute it asynchronously, and persist search results. Similar to SuiteAnalytics persisted search functionality, this capability is useful for searches across high volumes of data. You can create a task.SearchTask Object using task.create(options) .
---------------------------	--

Use the `SearchTask` Object to do the following:

- Export the results of a search into a CSV file in the file cabinet.
- Set the search and file ID values.
- Use `SearchTask.submit()` to submit the search task to the NetSuite task queue.
- Use the properties of `task.SearchTaskStatus` object to get the status of a search task.



Note: 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.

Supported Script Types	Server-side scripts
Module	N/task Module
Since	Version 2017 Release 1

Syntax

```
//Add additional code
...
var searchTask = task.create({
    taskType: task.TaskType.SEARCH
});
searchTask.savedSearchId = 51;
var path = 'ExportFolder/export.csv';
scriptTask.filePath = path;
var searchTaskId = searchTask.submit();
...
//Add additional code
```

SearchTask.submit()



Important: This is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any changes may impact the feature's operation with the NetSuite application. NetSuite warranties and product service levels shall not apply to the feature or the impact of the feature on other portions of the NetSuite application. NetSuite may review and monitor the performance and use of the feature.

Method Description	Directs NetSuite to initiate the asynchronous search task and return a unique ID for the task. Use <code>task.SearchTaskStatus</code> to view the status of a submitted task.
Returns	The task id as a string
Supported Script Types	Server-side scripts
Governance	5 units
Module	N/task Module
Since	Version 2017 Release 1

Errors

Error Code	Message	Thrown If
FAILED_TO_SUBMIT_JOB_REQUEST_1	Failed to submit job request: {reason}	Task cannot be submitted.
SSS_MISSING_REQD_ARGUMENT	{1}: Missing a required argument: {2}	A required parameter is missing.
YOU_DO_NOT_HAVE_ACCESS_TO_THE_MEDIA_ITEM_YOU_SELECTED	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	That record does not exist.{1}	The file Object references a file that doesn't exist.
MUST_IDENTIFY_A_FILE	A file must be specified: {1}	The path specifies a folder and not a 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 [N/task Module Script Sample](#).

```
//Add additional code
...
var searchTriggerTask = searchTask.submit();
...
//Add additional code
```

SearchTask.savedSearchID

 **Important:** This is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any changes may impact the feature's operation with the NetSuite application. NetSuite warranties and product service levels shall not apply to the feature or the impact of the feature on other portions of the NetSuite application. NetSuite may review and monitor the performance and use of the feature.

Property Description	ID of the saved search to be executed during the task.
Type	The saved search ID as a number
Module	N/task Module
Since	Version 2017 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/task Module Script Sample](#).

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

```
searchTask.savedSearchId = 51;
...
//Add additional code
```

SearchTask.fileId

⚠️ Important: This is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any changes may impact the feature's operation with the NetSuite application. NetSuite warranties and product service levels shall not apply to the feature or the impact of the feature on other portions of the NetSuite application. NetSuite may review and monitor the performance and use of the feature.

Property Description	ID of the CSV file to export search results into.
Type	The CSV file ID as a number
Module	N/task Module
Since	Version 2017 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
searchTask.fileId = 18;
...
//Add additional code
```

SearchTask.filePath

⚠️ Important: This is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any changes may impact the feature's operation with the NetSuite application. NetSuite warranties and product service levels shall not apply to the feature or the impact of the feature on other portions of the NetSuite application. NetSuite may review and monitor the performance and use of the feature.

Property Description	Path of the CSV file to export search results into.
Type	The CSV file path as a string
Module	N/task Module

Since	Version 2017 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
searchTask.filePath= 'ExportFolder/export.csv'
...
//Add additional code
```

task.SearchTaskStatus



Important: This is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any changes may impact the feature's operation with the NetSuite application. NetSuite warranties and product service levels shall not apply to the feature or the impact of the feature on other portions of the NetSuite application. NetSuite may review and monitor the performance and use of the feature.

Object Description	Encapsulates the status of an asynchronous search task (<code>task.SearchTask</code>) placed into the NetSuite task queue. To initiate the task and retrieve the task id, use SearchTask.submit() .
Supported Script Types	Server-side scripts
Module	N/task Module
Since	Version 2017 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/task Module Script Sample](#).

```
//Add additional code
...
var searchTaskStatus = task.checkStatus({
    searchTaskId:51
});
if (searchTaskStatus.status === task.TaskStatus.FAILED)
...
```

```
//Add additional code
```

SearchTaskStatus.fileId



Important: This is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any changes may impact the feature's operation with the NetSuite application. NetSuite warranties and product service levels shall not apply to the feature or the impact of the feature on other portions of the NetSuite application. NetSuite may review and monitor the performance and use of the feature.

Property Description	ID of CSV file into which search results are exported.
Type	CSV file id as a number
Module	N/task Module
Since	Version 2017 Release 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 [N/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



Important: This is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any changes may impact the feature's operation with the NetSuite application. NetSuite warranties and product service levels shall not apply to the feature or the impact of the feature on other portions of the NetSuite application. NetSuite may review and monitor the performance and use of the feature.

Property Description	The ID of the saved search executed during the task.
----------------------	--

Type	The search ID as a number
Module	N/task Module
Since	Version 2017 Release 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 [N/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

 **Important:** This is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any changes may impact the feature's operation with the NetSuite application. NetSuite warranties and product service levels shall not apply to the feature or the impact of the feature on other portions of the NetSuite application. NetSuite may review and monitor the performance and use of the feature.

Property Description	Status for an asynchronous search placed in the NetSuite task queue by <code>SearchTask.submit()</code> . Returns a <code>task.TaskStatus</code> enum value.
Type	<code>task.TaskStatus</code>
Module	N/task Module
Since	Version 2017 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
log.audit({
    title: 'Status',
    details: summary.status
});

...
//Add additional code
```

SearchTaskStatus.taskId



Important: This is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any changes may impact the feature's operation with the NetSuite application. NetSuite warranties and product service levels shall not apply to the feature or the impact of the feature on other portions of the NetSuite application. NetSuite may review and monitor the performance and use of the feature.

Property Description	ID of the <code>task.SearchTask</code> Object. Use <code>SearchTask.submit()</code> to return this ID.
Type	number
Module	N/task Module
Since	Version 2017 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
var searchTaskId = searchTask.submit();
...
//Add additional code
```

task.WorkflowTriggerTask

Object Description	Encapsulates all the properties required to asynchronously initiate a workflow. Use the <code>WorkflowTriggerTask</code> Object to create a task that initiates an instance of the specified workflow. The task is placed in the scheduling queue, and the workflow instance is initiated after the task reaches the top of the queue. To use the <code>WorkflowTriggerTask</code> Object:
	<ul style="list-style-type: none"> ■ Use <code>task.create(options)</code> to create the <code>WorkflowTriggerTask</code> Object. ■ Use <code>WorkflowTriggerTask.recordType</code> to set the record type of the workflow base record. ■ Use <code>WorkflowTriggerTask.recordId</code> to set the internal ID of the base record for the workflow. ■ Use <code>WorkflowTriggerTask.workflowId</code> to set the internal ID of the workflow that you want to run on the record specified by the <code>recordId</code>. ■ Optionally, use <code>WorkflowTriggerTask.params</code> to specify default values for workflow fields. ■ Use <code>WorkflowTriggerTask.submit()</code> to submit the asynchronous workflow initiation task to the NetSuite task queue. ■ Use the properties for the <code>WorkflowTriggerTaskStatus.status</code> object to get the status of the workflow execution.
	Use the following guidelines with the <code>WorkflowTriggerTask</code> Object:
	<ul style="list-style-type: none"> ■ <code>WorkflowTriggerTask.submit()</code> does not successfully place a workflow task in the scheduling queue if an identical instance of that workflow, with the same <code>recordType</code>, <code>recordId</code>, and <code>workflowId</code>, is currently executing or already in the scheduling queue.
	For a complete list of this object's methods and properties, see WorkflowTriggerTask Object Members .
Supported Script Types	Server-side scripts
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

Method Description	Directs NetSuite to place the asynchronous workflow initiation task into the NetSuite scheduling queue and returns a unique ID for the task. Use WorkflowTriggerTaskStatus.status to view the status of a submitted task.
Returns	the task id as a string
Supported Script Types	Server-side scripts
Governance	20 units
Module	N/task Module
Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
var workflowTriggerTask = workflowTask.submit();
...
//Add additional code
```

WorkflowTriggerTask.recordType

Property Description	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.
Type	string
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

```
//Add additional code
```

```

...
workflowTask.recordType = 'customer';
...
//Add additional code

```

WorkflowTriggerTask.recordId

Property Description	Internal ID of the base record. For example, 55 or 124.
Type	number
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

```

//Add additional code
...
workflowTask.recordId = 107;
...
//Add additional code

```

WorkflowTriggerTask.workflowId

Property Description	Internal ID (as a number), or script ID (as a string), for the workflow definition. This is the ID that appears in the ID field on the Workflow Definition Page .
Type	number string
Module	N/task Module
Since	Version 2015 Release 2

Syntax



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

```

//Add additional code
...
workflowTask.workflowId = 3;
...
//Add additional code

```

WorkflowTriggerTask.params

Property Description	Object that contains key/value pairs to set default values on fields specific to the workflow. These can include fields on the Workflow Definition Page or workflow and state Workflow Custom Fields .
Type	Object
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
task.params = {context: portlet}
...
//Add additional code
```

task.WorkflowTriggerTaskStatus

Object Description	Encapsulates the status of an asynchronous workflow initiation task placed into the NetSuite task queue by WorkflowTriggerTask.submit() . Use task.checkStatus(options) with the unique ID for the asynchronous workflow initiation task to get the WorkflowTriggerTaskStatus object. For a complete list of this object's properties, see WorkflowTriggerTaskStatus Object Members .
Supported Script Types	Server-side scripts
Module	N/task Module
Since	Version 2015 Release 2

Syntax

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

```
//Add additional code
...
var workflowTaskStatus = task.checkStatus(taskId);
if (workflowTaskStatus.status === task.TaskStatus.FAILED)
...
//Add additional code
```

WorkflowTriggerTaskStatus.status

Property Description	Status for an asynchronous workflow placed in the NetSuite task queue by <code>WorkflowTriggerTask.submit()</code> . Returns a <code>task.TaskStatus</code> enum value.
Type	<code>task.TaskStatus</code>
Module	N/task Module
Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
log.audit('Status', summary.status);

...
//Add additional code
```

task.create(options)

Method Description	Creates an object for a specific task type and returns the task object. Use with the N/task Module to create a task to schedule scripts, run map/reduce scripts, import CSV files, merge duplicate records, initiate asynchronous searches, or execute asynchronous workflows.
	 Important: Initiating asynchronous searches is available as a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any changes may impact the feature's operation with the NetSuite application. NetSuite warranties and product service levels shall not apply to the feature or the impact of the feature on other portions of the NetSuite application. NetSuite may review and monitor the performance and use of the feature.
Returns	<code>task.ScheduledScriptTask</code> <code>task.MapReduceScriptTask</code> <code>task.CsvImportTask</code> <code>task.EntityDeduplicationTask</code> <code>task.WorkflowTriggerTask</code> <code>task.SearchTask</code> (BETA)
Supported Script Types	Server-side scripts
Governance	None
Module	N/task Module

Since	Version 2015 Release 2
-------	------------------------

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.taskType	task.TaskType	Required	The type of task object to create. Use the task.TaskType enum to set the value.	Version 2015 Release 2
options.scriptId	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 ScheduledScriptTask.scriptId or MapReduceScriptTask.scriptId property. Only applicable when <code>taskType</code> is set to SCHEDULED_SCRIPT or MAP_REDUCE .	2016.2
options.deploymentId	number string	Optional	The internal ID (as a number) or script ID (as a string) of the script deployment record. This parameter sets the value for the ScheduledScriptTask.deploymentId or MapReduceScriptTask.deploymentId property. Only applicable when <code>taskType</code> is set to SCHEDULED_SCRIPT or MAP_REDUCE .	2016.2
options.params	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 Creating Script Parameters Overview . For Workflow tasks, keys can include fields on the Workflow Definition Page or workflow and state Workflow Custom Fields . This parameter sets the value for the ScheduledScriptTask.params , MapReduceScriptTask.params or WorkflowTriggerTask.params property. Only applicable when <code>taskType</code> is set to	2016.2

Parameter	Type	Required / Optional	Description	Since
			SCHEDULED_SCRIPT, MAP_REDUCE or WORKFLOW_TRIGGER.	
options.importFile	file.File string	Optional	A CSV file to import. Use a file.File object or a string that represents the CSV text to be imported. This parameter sets the value for the CsvImportTask.importFile property. Only applicable when taskType is set to CSV_IMPORT .	2016.2
options.mappingId	number string	Optional	The internal ID (as a number) or script ID (as a string) of a saved import map that you created when you ran the Import Assistant. See task.CsvImportTask.mappingId . This parameter sets the value for the CsvImportTask.mappingId property. Only applicable when taskType is set to CSV_IMPORT .	2016.2
options.queueId	number	Optional	Overrides the Queue Number property under Advanced Options on the Import Options page of the Import Assistant. Use this property to programmatically select an import queue and improve performance during the import.	2016.2
<p>Note: This property is only available if you have a SuiteCloud Plus license. For more information about using multiple queues when importing CSV files, see the help topics Queue Number and Using Multiple Threads and Multiple Queues to Run CSV Import Jobs.</p> <p>This parameter sets the value for the CsvImportTask.queueId property. Only applicable when taskType is set to CSV_IMPORT.</p>				
options.name	string	Optional	The 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	2016.2

Parameter	Type	Required / Optional	Description	Since
			the CSV Import Job Status page. This parameter sets the value for the CsvImportTask.name property. Only applicable when <code>taskType</code> is set to CSV_IMPORT .	
options.linkedFiles	Object	Optional	A map of key/value pairs that sets the data to be imported in a linked file for a multi-file import job, by referencing a file in the file cabinet or the raw CSV data to import. The key is the internal ID of the record sublist for which data is being imported and the value is either a file.File object or the raw CSV data to import. You can assign multiple types of values to the <code>linkedFiles</code> property. This parameter sets the value for the CsvImportTask.linkedFiles property. Only applicable when <code>taskType</code> is set to CSV_IMPORT .	2016.2
options.entityType	string	Optional	Sets the type of entity on which you want to merge duplicate records. This parameter sets the value for the EntityDeduplicationTask.entityType property. Only applicable when <code>taskType</code> is set to ENTITY_DEDUPLICATION . Use the task.DedupeEntityType enum to set the value.	2016.2
options.masterRecordId	number	Optional	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.	2016.2

Parameter	Type	Required / Optional	Description	Since
			<p> Important: You must also select <code>SELECT_BY_ID</code> for the <code>EntityDeduplicationTask.masterSelectionMode</code> property, or NetSuite ignores this setting.</p> <p>This parameter sets the value for the <code>EntityDeduplicationTask.masterRecordId</code> property. Only applicable when <code>taskType</code> is set to <code>ENTITY_DEDUPLICATION</code>.</p>	
<code>options.masterSelectionMode</code>	<code>string</code>	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. 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. Only applicable when <code>taskType</code> is set to <code>ENTITY_DEDUPLICATION</code>. Use the <code>task.MasterSelectionMode</code> enum to set the value.</p>	2016.2
<code>options.dedupeMode</code>	<code>string</code>	Optional	<p>Sets the mode in which to merge or delete duplicate records.</p> <p>This parameter sets the value for the <code>EntityDeduplicationTask.dedupeMode</code> property. Only applicable when <code>taskType</code> is set to <code>ENTITY_DEDUPLICATION</code>. Use the <code>task.DedupeMode</code> enum to set the value.</p>	2016.2
<code>options.recordIds</code>	<code>number[]</code>	Optional	<p>The number array of record internal IDs to perform the merge or delete operation on.</p> <p>You can use the <code>search.duplicates(options)</code> method to identify duplicate records or create an array with record internal IDs.</p> <p>This parameter sets the value for the <code>EntityDeduplicationTask.recordIds</code> property. Only applicable when <code>taskType</code> is set to <code>ENTITY_DEDUPLICATION</code>.</p>	2016.2

Parameter	Type	Required / Optional	Description	Since
options.recordType	string	Optional	The record type of the workflow definition base record, such as customer, salesorder, or lead. In the Workflow Manager, this is the record type that is specified in the Record Type field. This parameter sets the value for the WorkflowTriggerTask.recordType property. Only applicable when <code>taskType</code> is set to <code>WORKFLOW_TRIGGER</code> .	2016.2
options.recordId	number	Optional	The internal ID of the base record. This parameter sets the value for the WorkflowTriggerTask.recordId property. Only applicable when <code>taskType</code> is set to <code>WORKFLOW_TRIGGER</code> .	2016.2
options.workflowId	number string	Optional	The internal ID (as a number) or script ID (as a string) for the workflow definition. This is the ID that appears in the ID field on the Workflow Definition Page . This parameter sets the value for the WorkflowTriggerTask.workflowId property. Only applicable when <code>taskType</code> is set to <code>WORKFLOW_TRIGGER</code> .	2016.2
options.savedSearchId	number	Optional	The ID of the saved search to be executed during the task.	2017.1
options.fileId (BETA)	string	Optional	The ID of the CSV file to export search results to. See N/file Module .	2017.1
options.filePath (BETA)	number	Optional	<p>Note: 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> Path of the CSV file to export search results to. See N/file Module .	2017.1

Parameter	Type	Required / Optional	Description	Since
			<p>Note: 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 [N/task Module Script Sample](#).

```
//Add additional code
...
var mrTask = task.create({
    taskType: task.TaskType.MAP_REDUCE,
    scriptId: 34,
    deploymentId: 1,
    params: {doSomething: true}
});
...
//Add additional code
```

task.checkStatus(options)

Method Description	Returns a task status object associated with a specific task ID.
Returns	<code>task.ScheduledScriptTaskStatus</code> <code>task.MapReduceScriptTaskStatus</code> <code>task.CsvImportTaskStatus</code> <code>task.EntityDeduplicationTaskStatus</code> <code>task.SearchTaskStatus</code> <code>task.WorkflowTriggerTaskStatus</code>
Supported Script Types	Server-side scripts
Governance	None
Module	N/task Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.taskId</code>	<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> .	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
	nTask 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 [N/task Module Script Sample](#).

```
//Add additional code
...
var taskStatus = task.checkStatus(mrTaskId);
...
//Add additional code
```

task.TaskType

Enum Description	Enumeration that holds the string values for the types of task objects supported by the N/task Module , that you can create with <code>task.create(options)</code> . Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Module	N/task Module
Since	Version 2015 Release 2

Values

- SCHEDULED_SCRIPT
- MAP_REDUCE
- CSV_IMPORT
- ENTITY_DEDUPLICATION
- SEARCH
-
- Note:** The search task type is a beta feature.
- WORKFLOW_TRIGGER

Syntax

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

```
//Add additional code
```

```

...
var mrTask = task.create({
    taskType: task.TaskType.MAP_REDUCE
});
...
//Add additional code

```

task.TaskStatus

Enum Description	<p>Enumeration that holds the string values for the possible status of tasks created and submitted with the N/task Module. The following properties hold a value for <code>task.taskStatus</code>:</p> <ul style="list-style-type: none"> ▪ <code>ScheduledScriptTaskStatus.status</code> ▪ <code>MapReduceScriptTaskStatus.status</code> ▪ <code>CsvImportTaskStatus.status</code> ▪ <code>EntityDeduplicationTaskStatus.status</code> ▪ <code>SearchTaskStatus.status</code> (BETA) ▪ <code>WorkflowTriggerTaskStatus.status</code> <p>Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Module	N/task Module
Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```

//Add additional code
...
if (status == task.TaskStatus.COMPLETE || status == task.TaskStatus.FAILED)
...
//Add additional code

```

task.MasterSelectionMode

Enum Description	Enumeration that holds the string values for supported master selection modes when merging duplicate records with <code>task.EntityDeduplicationTask</code> . Use this enum for the <code>EntityDeduplicationTask.masterSelectionMode</code> property. For more information about these values, see the help topic Merging or Deleting Duplicate Records .
Module	N/task Module
Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
dedupeTask.masterSelectionMode = task.MasterSelectionMode.MOST_RECENT_ACTIVITY;
...
//Add additional code
```

task.DedupeMode

Enum Description	Enumeration that holds the string values for the available deduplication modes when merging duplicate records with <code>task.EntityDeduplicationTask</code> . Use this enum for the <code>EntityDeduplicationTask.dedupeMode</code> property.
Module	N/task Module

Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
dedupeTask.dedupeMode = task.DedupeMode.MERGE;
...
//Add additional code
```

task.DedupeEntityType

Enum Description	Enumeration that holds the string values for entity types for which you can merge duplicate records with task.EntityDeduplicationTask . Use this enum for the EntityDeduplicationTask.entityType .
	<p>i Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Module	N/task Module
Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
dedupeTask.entityType = task.DedupeEntityType.CUSTOMER;
...
//Add additional code
```

task.MapReduceStage

Enum Description	Enumeration that holds the string values for possible stages in <code>task.MapReduceScriptTask</code> for a map/reduce script. This enum is returned by <code>MapReduceScriptTaskStatus.stage</code> .
Module	N/task Module
Since	Version 2015 Release 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 [N/task Module Script Sample](#).

```
//Add additional code
...
if (summary.stage === task.MapReduceStage.SUMMARIZE)
...
//Add additional code
```

N/transaction Module

Load the transaction module to void transactions.

When you void a transaction, the total and all the line items for the transaction are set to zero. The transaction is not removed from the system. NetSuite supports two types of voids: direct voids and voids by reversing journal. For additional information, see the help topic [Voiding, Deleting, or Closing Transactions](#).

The type of void performed with your script depends on the targeted account's preference settings:

- If the Using Reversing Journals preference is disabled, a **direct void** is performed.
- If the Using Reversing Journals preference is enabled, a **void by reversing journal** is performed.



Important: After you successfully void a transaction, you can no longer make changes to the transaction that impact the general ledger.



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

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

N/transaction Module Members

Member Type	Name	Return Type / Value Type	Description
Method	transaction.void(options)	number	Voids a transaction record.
	transaction.void.promise(options)	number	Voids a transaction record asynchronously.
Enum	transaction.Type	enum	Enumeration that holds the string values for supported record types.

N/transaction Module Script Sample



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

The following examples voids a transaction.

```
/** 
 * @NApiVersion 2.x
 */
// This example shows how to void a transaction
require(['N/transaction', 'N/config', 'N/record'],
    function(transaction, config, record) {
        function voidSalesOrder() {
            var accountingConfig = config.load({
                type: config.Type.ACOUNTING_PREFERENCES
            });
            accountingConfig.setValue({
                fieldId: 'REVERSALVOIDING',
                value: false
            });
            accountingConfig.save();
        }
    }
);
```

```

var salesOrderObj = record.create({
    type: 'salesorder',
    isDynamic: false
});
salesOrderObj.setValue({
    fieldId: 'entity',
    value: 107
});
salesOrderObj.setSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    value: 233,
    line: 0
});
salesOrderObj.setSublistValue({
    sublistId: 'item',
    fieldId: 'amount',
    value: 1,
    line: 0
});
var salesOrderId = salesOrderObj.save();
var voidSalesOrderId = transaction,void({
    type: record.Type.SALES_ORDER,
    id: salesOrderId
});
var salesOrder = record.load({
    type: 'salesorder',
    id: voidSalesOrderId
});
// memo should be 'VOID'
var memo = salesOrder.getValue({
    fieldId: 'memo'
});
}
voidSalesOrder();
});

```

transaction.void(options)

Method Description	Method used to void a transaction record object and return an id that indicates the type of void performed. The type of void performed depends on the targeted account's preference settings.
Returns	<p>An ID returned as a number.</p> <ul style="list-style-type: none"> ■ If a direct void is performed, returns the ID of the record voided. ■ If a void by reversing journal is performed, returns the ID of the newly created voiding journal.

Supported Script Types	All client and server-side scripts
Governance	10 units
Module	N/transaction Module
Since	Version 2015 Release 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.	Version 2015 Release 2
options.type	string	required	Internal ID of the type of transaction record to void	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
INVALID_RECORD_TYPE		The <code>type</code> argument passed is not valid or the record type is not voidable.
THAT_RECORD_DOES_NOT_EXIST		The <code>id</code> argument passed is not valid.
SSS_MISSING_REQD_ARGUMENT		The <code>type</code> or <code>id</code> 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/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)

Method Description	Method used to void a transaction record object asynchronously and return an id that indicates the type of void performed.
--------------------	--

	<p>The type of void performed depends on the targeted account's preference settings.</p> <p>Important: After you void a transaction, you cannot make changes to the transaction that impact the general ledger.</p> <p>Note: For information about the parameters and errors thrown for this method, see transaction.void(options). For additional information on promises, see Promise object.</p>
Returns	<p>An id returned as a number.</p> <ul style="list-style-type: none"> ■ If a direct void is performed, returns the ID of the record voided. ■ If a void by reversing journal is performed, returns the ID of the newly created voiding journal.
Synchronous Version	transaction.void(options)
Supported Script Types	All client-side scripts
Governance	10 units
Module	N/transaction Module
Since	Version 2015 Release 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 voidSalesOrderId = transaction.void.promise({
    type: record.Type.SALES_ORDER,
    id: salesOrderId
});
...
//Add additional code
```

transaction.Type

Enum Description	<p>Enumeration that holds the string values for supported transaction record types.</p> <p>Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Module	N/transaction Module
Since	Version 2015 Release 2

Values

Transaction Record	Supported Void Type
CASH_REFUND	Direct Void
CASH_SALE	Direct Void
CHECK	Void by Reversing Journal
CREDIT_MEMO	Direct Void
CUSTOMER_DEPOSIT	Direct Void
CUSTOMER_PAYMENT	Direct Void
CUSTOMER_REFUND	Direct Void and Void by Reversing Journal
ESTIMATE_QUOTE	Direct Void
EXPENSE_REPORT	Direct Void
INTERCOMPANY_JOURNAL_ENTRY	Direct Void
INVOICE	Direct Void
JOURNAL_ENTRY	Direct Void
PAYCHECK_JOURNAL	Direct Void
RETURN_AUTHORIZATION	Direct Void
SALES_ORDER	Direct Void
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

Syntax



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 voidSalesOrderId = transaction.void({
    type: transaction.Type.SALES_ORDER,
    id: salesOrderId
});
...
//Add additional code
```

N/ui/dialog Module

Load the dialog module to create a modal dialog that persists until a button on the dialog is pressed.



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

- N/ui/dialog Module Members
- N/ui/dialog Module Script Samples

N/ui/dialog Module Members

Member Type	Name	Property Type / Method Return Type	Description
Method	dialog.alert(options)	Promise	Creates an Alert dialog with an OK button.
	dialog.confirm(options)	Promise	Creates a Confirm dialog with OK and Cancel buttons.
	dialog.create(options)	Promise	Creates a dialog with specified buttons.

N/ui/dialog Module Script Samples

The following example shows how to create an Alert dialog:

```
/** @NApiVersion 2.x */
require(['N/ui/dialog'],
    function(dialog) {
        var options = {
            title: "I am an Alert",
            message: "Press OK"
        };
        function success(result) {
            console.log("Success with value " + result);
        }
        function failure(reason) {
            console.log("Failure: " + reason);
        }

        dialog.alert(options).then(success).catch(failure);
    });
});
```

The following sample shows how to create a Confirmation dialog:

```
/** @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 sample shows how to create a dialog with buttons:

```

/**
 * @NApiVersion 2.x
 */
require(['N/ui/dialog'],
    function(dialog) {
        var button1 = {
            label: 'I am A',
            value: 1
        };
        var button2 = {
            label: 'I am B',
            value: 2
        };
        var button3 = {
            label: 'I am C',
            value: 3
        };
        var options = {
            title: 'Alphabet Test',
            message: 'Which One?',
            buttons: [button1, button2, button3]
        };

        function success(result) {
            console.log("Success with value " + result);
        }
        function failure(reason) {
            console.log("Failure: " + reason);
        }
        dialog.create(options).then(success).catch(failure);
    });

```

dialog.alert(options)

Method Description	Creates an Alert dialog with an OK button.
Returns	Promise object. To run a callback function when the OK button is pressed, pass a function to the then portion of the Promise object. When the OK button is pressed, true is passed to the callback.
Supported Script Types	Client scripts

Governance	None
Module	N/ui/dialog Module
Since	Version 2016 Release 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.	Version 2016 Release 1
options.message	string	optional	The content of the alert dialog. This value defaults to an empty string.	Version 2016 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/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: 'Alert',
    message: 'Click OK to continue.'
}).then(success).catch(failure);
...
//Add additional code
```

dialog.confirm(options)

Method Description	Creates a Confirm dialog with OK and Cancel buttons.
Returns	Promise object. To run a callback function when the OK button is pressed, pass a function to the then portion of the Promise object. The value of the pressed button, where OK is true and Cancel is false , is passed to the callback.
Supported Script Types	Client scripts
Governance	None
Module	N/ui/dialog Module

Since	Version 2016 Release 1
-------	------------------------

Parameters

i 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.	Version 2016 Release 1
options.message	string	optional	The content of the confirmation dialog. This value defaults to an empty string.	Version 2016 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/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
```

dialog.create(options)

Method Description	Creates a dialog with specified buttons.
Returns	Promise object. To run a callback function when a button is pressed, pass a function to the <code>then</code> portion of the Promise object. The value of the button pressed is passed to the callback.

Supported Script Types	Client scripts
Governance	None
Module	N/ui/dialog Module
Since	Version 2016 Release 1

Parameters

i Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.buttons	string[]	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.	Version 2016 Release 1
options.title	string	optional	The dialog title. This value defaults to an empty string.	Version 2016 Release 1
options.message	string	optional	The content of the dialog. This value defaults to an empty string.	Version 2016 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/ui/dialog Module Script Samples](#).

```
//Add additional code
...
var options = {
    title: 'Dialog',
    message: 'Click a button to continue.',
    buttons: [
        { label: '1', value: 1 },
        { label: '2', value: 2 },
        { label: '3', value: 3 }
    ];
};

function success(result) { console.log('Success with value: ' + result) }
function failure(reason) { console.log('Failure: ' + reason) }

dialog.create(options).then(success).catch(failure);
...
//Add additional code
```

N/ui/message module

Load the message module to display a message at the top of the screen under the menu bar.

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

- N/ui/message Members
- Message Object Members
- N/ui/message Module Script Sample

N/ui/message Members

Member Type	Name	Return Type / Value Type	Description
Object	message.Message	void	Encapsulates the Message object that gets created when calling the create method.
Method	message.create(options)	Message	Creates a message that can be displayed or hidden near the top of the page.
Enum	message.Type	enum	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	Description
Method	Message.hide()	void	Hides the message.
	Message.show()	void	Shows the message.

N/ui/message Module Script Sample

The following example shows how to create a confirmation message:

```
/** 
 * @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(); // will stay up until hide is called.
});

```

message.Message

Object Description	Encapsulates the Message object that gets created when calling the create method. For a complete list of this object's methods, see Message Object Members .
Supported Script Types	Client scripts
Module	N/ui/message module
Since	Version 2016 Release 1

Message.hide()

Method Description	Hides the message.
Returns	void
Supported Script Types	Client scripts
Governance	None
Module	N/ui/message module
Since	Version 2016 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/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()

Method Description	Shows the message.
Returns	void
Supported Script Types	Client scripts
Governance	None
Module	N/ui/message module
Since	Version 2016 Release 1

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.duration	int	optional	The amount of time, in milliseconds, to show the message. The default is 0, which shows the message until Message.hide() is called.	Version 2016 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/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)

Method Description	Creates a message that can be displayed or hidden near the top of the page.
Returns	<code>message.Message</code> .
Supported Script Types	Client scripts
Governance	None
Module	N/ui/message module
Since	Version 2016 Release 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.	Version 2016 Release 1
<code>options.title</code>	<code>string</code>	optional	The message title. This value defaults to an empty string.	Version 2016 Release 1
<code>options.message</code>	<code>string</code>	optional	The content of the message. This value defaults to an empty string.	Version 2016 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/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

Enum Description	Indicates the type of message to display, which specifies the background color of the message and other message indicators.
-------------------------	---

Module	N/ui/message module
Since	Version 2016 Release 1

Values

Value	Color
CONFIRMATION	A green background with a checkmark icon.
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

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:** 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 Sample

N/ui/serverWidget Module Members

Member Type	Name	Return Type / Value Type	Description
Object	serverWidget.Assistant	Object	Encapsulates a scriptable, multi-step NetSuite assistant.
	serverWidget.AssistantStep	Object	Encapsulates a step within a custom NetSuite assistant.
	serverWidget.Button	Object	Encapsulates a button that appears in a UI object.
	serverWidget.Field	Object	Encapsulates a NetSuite field.
	serverWidget.FieldGroup	Object	Encapsulates a field group.
	serverWidget.Form	Object	Encapsulates a NetSuite form.
	serverWidget.List	Object	Encapsulates a list.
	serverWidget.ListColumn	Object	Encapsulates list columns.
	serverWidget.Sublist	Object	Encapsulates a NetSuite sublist.
Method	serverWidget.createAssistant(options)	serverWidget.Assistant	Creates and returns a new assistant object.
	serverWidget.createForm(options)	serverWidget.Form	Creates and returns a new form object.
	serverWidget.createList(options)	serverWidget.List	Instantiates a List object (specifying the title, and whether to hide the navigation bar)
Enum	serverWidget.AssistantSubmitAction	string (read-only)	Holds the string values for submit actions performed by the user.
	serverWidget.FieldBreakType	string (read-only)	Holds the string values for supported field break types. This enum is used to set the value of the Field.updateBreakType(options) property.
	serverWidget.FieldDisplayType	string (read-only)	Holds the string values for supported field display types. This enum is used to set the value of the Field.updateDisplayType(options) property.

Member Type	Name	Return Type / Value Type	Description
	serverWidget.FieldLayoutType	string (read-only)	Holds the string values for the supported types of field layouts. This enum is used to set the value of the Field.updateLayoutType(options) property.
	serverWidgetFieldType	string (read-only)	Holds the values for supported field types. This enum is used to set the value of the Field.type property.
	serverWidgetFormPageLinkType	string (read-only)	Holds the string values for supported page link types on a form. This enum is used to set the value of the type parameter for Form.addPageLink(options) .
	serverWidgetLayoutJustification	string (read-only)	Holds the string values for supported justification layouts. This enum is used to set the value of the align parameter when List.addColumn(options) is called.
	serverWidgetListStyle	string (read-only)	Holds the string values for supported list styles. This enum is used to set the value of the List.style property.
	serverWidgetLayoutJustification	string (read-only)	Holds the string values for supported sublist display types. This enum is used to set the value of the Sublist.displayType property.
	serverWidgetSublistType	string (read-only)	Holds the string values for valid sublist types. This enum is used to define the type parameter when Form.addSublist(options) is called

Assistant Object Members

The following members are called on the [serverWidget.Assistant](#) object.

Member Type	Name	Property Type / Method Return Type	Description
Method	Assistant.addField(options)	serverWidget.Field	Adds a field to an assistant.
	Assistant.addFieldGroup(options)	serverWidget.FieldGroup	Adds a field group to an assistant.

Member Type	Name	Property Type / Method Return Type	Description
	Assistant.addStep(options)	serverWidget.AssistantStep	Adds a step to an assistant.
	Assistant.addSublist(options)	serverWidget.Sublist	Adds a sublist to an assistant.
	Assistant.getField(options)	serverWidget.Field	Gets a field object.
	Assistant.getFieldGroup(options)	serverWidget.FieldGroup	Gets a field group object.
	Assistant.getFieldGroupIds()	string	Gets all the field group IDs in an assistant.
	Assistant.getFieldIds()	string	Gets all the field IDs in an assistant.
	Assistant.getFieldIdsByFieldGroup(fieldGroup)	string[]	Gets all field IDs in the assistant field group.
	Assistant.getLastAction()	string	Gets the last action submitted by the user.
	Assistant.getLastStep()	serverWidget.AssistantStep	Gets the step that the last submitted action came from.
	Assistant.getNextStep()	serverWidget.AssistantStep	Gets the next step prompted by the assistant.
	Assistant.getStep(options)	serverWidget.AssistantStep	Returns a step in an assistant.
	Assistant.getStepCount()	serverWidget.AssistantStep	Gets the total count of steps in the assistant.
	Assistant.getSteps()	serverWidget.AssistantStep[]	Gets all the steps in the assistant.
	Assistant.getSublist(options)	serverWidget.Sublist	Get a Sublist object from its ID.
	Assistant.getSublistIds()	string	Gets all the sublist IDs in an assistant.
	Assistant.hasErrorHtml()	boolean true false	Indicates whether the assistant threw an error.
	Assistant.isFinished()	boolean true false	Sets the status of the assistant. If set to true, the assistant is finished.
	Assistant.sendRedirect(options)	void	Manages redirects in an assistant.
	Assistant.setSplash(options)	void	Define a splash message.
	Assistant.updateDefaultValues(values)	void	Sets the default values of an array of fields

Member Type	Name	Property Type / Method Return Type	Description
			that are specific to the assistant.
Property	Assistant.clientScriptFileId	number	The file cabinet ID of client script file to be used in this assistant.
	Assistant.clientScriptModulePath	string	The relative path to the client script file to be used in this assistant.
	Assistant.currentStep	serverWidget.AssistantStep	Identifies the current step.
	Assistant.errorHtml	string	The error message text.
	Assistant.finishedHtml	string	The text displayed after an assistant is finished.
	Assistant.hideAddToShortcutsLink	boolean true false	Indicates whether the Add to Shortcuts Link is displayed in the UI.
	Assistant.hideStepNumber	boolean true false	Indicates whether the current and total step numbers are displayed in the UI.
	Assistant.isNotOrdered	boolean true false	Indicates whether assistant steps are ordered or unordered.
	Assistant.title	string	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	Description
Method	AssistantStep.getFieldIds()	string	Gets all the field IDs in an assistant step.
	AssistantStep.getLineCount(options)	number	Gets the number of lines previously entered by a user in a step.
	AssistantStep.getLineCount(options)	string	Gets all the field IDs in a list.
	AssistantStep.getSubmittedSublistIds()	string	Gets the IDs for all the sublist fields (line items) in a step.
	AssistantStep.getSublistValue(options)	string	Gets the current value of a sublist field (line item) in a step.

Member Type	Name	Return Type / Value Type	Description
	AssistantStep.getValue(options)	string	Gets the current value of a field.
Property	AssistantStep.helpText	string	The help text for a step.
	AssistantStep.id	string	The internal ID of the step.
	AssistantStep.label	string	The label for a step.
	AssistantStep.stepNumber	number	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	Description
Property	Button.isDisabled	boolean true false	Indicates whether a button is grayed-out and disabled.
	Button.isHidden	boolean true false	Indicates whether the button is hidden in the UI.
	Button.label	string	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	Description
Method	Field.addSelectOption(options)	void	Adds a select option to a dropdown list for a selectable field.
	Field.getSelectOptions(options)	object[]	Returns the internal ID and label of the options for a select field as name/value pairs.
	Field.setHelpText(options)	void	Sets the help text that appears in the field help popup.
	Field.updateBreakType(options)	serverWidget.Field	Updates the break type used to add a break in flow layout for the field.
	Field.updateDisplaySize(options)	serverWidget.Field	Updates the height and width for the field.
	Field.updateDisplayType(options)	serverWidget.Field	Updates the type of display for the field.

Member Type	Name	Return Type / Value Type	Description
	Field.updateLayoutType(options)	serverWidget.Field	Updates the layout type for the field.
Property	Field.alias	string	The alias used to set the field value.
	Field.defaultValue	string	The default value for the field.
	Field.id	string	The internal ID for the field.
	Field.isMandatory	boolean true false	Indicates whether the field is mandatory.
	Field.label	string	Gets or sets the label for the field.
	Field.linkText	string	The text displayed for a link in place of the URL.
	Field.maxLength	number	The maximum length, in characters, for the field.
	Field.padding	number	The number of empty vertical character spaces above the field.
	Field.richTextHeight	number	The height of a rich text field, in pixels.
	Field.richTextWidth	number	The width of a rich text field, in pixels.
	Field.type	string	The type of field.

FieldGroup Object Members

The following members are called on the `serverWidget.FieldGroup` object.

Member Type	Name	Property Type	Description
Property	FieldGroup.isBorderHidden	boolean true false	Indicates whether a border appears around the field group.
	FieldGroup.isCollapsible	boolean true false	Indicates whether the field group is collapsible.
	FieldGroup.isCollapsed	boolean true false	Indicates whether the field group is initially collapsed or expanded in the default view.
	FieldGroup.isSingleColumn	boolean true false	Indicates whether the field group is aligned.
	FieldGroup.label	string	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	Description
Method	Form.addButton(options)	serverWidget.Button	Adds a button to the form.
	Form.addCredentialField(options)	serverWidget.Field	Adds a field that store credentials in NetSuite for invoking services provided by third parties.
	Form.addField(options)	serverWidget.Field	Adds a field to the form.
	Form.addFieldGroup(options)	serverWidget.FieldGroup	Adds a group of fields to the form.
	Form.addPageLink(options)	void	Adds a link to a form.
	Form.addResetButton(options)	serverWidget.Button	Adds a reset button to a form that clears user input.
	Form.addSecretKeyField(options)	serverWidget.Field	Add a secret key field to the form.
	Form.addSublist(options)	serverWidget.Sublist	Adds a sublist to the form.
	Form.addSubmitButton(options)	serverWidget.Button	Adds a submit button to a form that saves user inputs.
	Form.addSubtab(options)	serverWidget.Tab	Adds a subtab to a form.
	Form.addTab(options)	serverWidget.Tab	Adds a tab to a form.
	Form.getButton(options)	serverWidget.Button	Returns a button by internal ID.
	Form.getField(options)	serverWidget.Field	Returns a field by internal ID.
	Form.getSublist(options)	serverWidget.Sublist	Returns a sublist by internal ID.
	Form.getSubtab(options)	serverWidget.Tab	Returns a subtab by internal ID.
	Form.getTab(options)	serverWidget.Tab	Returns a tab object from its internal ID.
	Form.getTabs()	serverWidget.Tab[]	Returns an array of all the tabs in a form.
	Form.insertField(options)	void	Inserts a field before another field within a form.
	Form.insertSublist(options)	serverWidget.Sublist	Inserts a sublist before another sublist on a form.

Member Type	Name	Property Type / Method Return Type	Description
	Form.insertSubtab(options)	serverWidget.Tab	Inserts a subtab before another subtab on a form.
	Form.insertTab(options)	serverWidget.Tab	Inserts a tab before another tab on a form.
	Form.removeButton(options)	void	Removes a button from a form.
	Form.updateDefaultValue s(options)	void	Sets the default values of many fields on a form.
Property	Form.clientScriptFileId	number	The file cabinet ID of client script file to be used in this form.
	Form.clientScriptModuleP ath	string	The relative path to the client script file to be used in this form.
	Form.title	string	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	Description
Method	List.addButton(options)	serverWidget.Button	Adds a button to a list.
	List.addColumn(options)	serverWidget.ListColumn	Adds a column to a list.
	List.addEditColumn(options)	serverWidget.ListColumn	Adds a column containing Edit or Edit/View links to a Suitelet or Portlet list.
	List.addPageLink(options)	void	Adds a link to a list.
	List.addRow(options)	void	Adds a single row to a list.
	List.addRows(options)	void	Adds multiple rows to a list.
Property	List.clientScriptFileId	number	The file cabinet ID of client script file to be used in this list.
	List.clientScriptModulePa th	string	The relative path to the client script file to be used in this list.
	List.style	string	Sets the display style for this list.

Member Type	Name	Property Type / Method Return Type	Description
	List.title	string	Sets the List title.

ListColumn Object Members

The following members are called on the `serverWidget.ListColumn` object.

Member Type	Name	Return Type / Value Type	Description
Method	ListColumn.addParamToURL(options)	serverWidget.ListColumn	Adds a URL parameter (optionally defined per row) to the list column's URL.
	ListColumn.setURL(options)	serverWidget.ListColumn	Sets the base URL for the list column.
Property	ListColumn.label	string	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	Description
Method	Sublist.addButton(options)	serverWidget.Button	Adds a button to a sublist.
	Sublist.addField(options)	serverWidget.Field	Add a field to a sublist.
	Sublist.addMarkAllButtons()	serverWidget.Button	Adds a Mark All or Unmark All button.
	Sublist.addRefreshButton()	serverWidget.Button	Adds a Reset button.
	Sublist.getField(options)	serverWidget.Field	Returns a Field object on a specified sublist.
	Sublist.getSublistValue(options)	serverWidget.Button	Adds a button to a sublist.
	Sublist.setSublistValue(options)	void	Sets the value of a sublist field.
	Sublist.updateTotallingFieldId(options)	serverWidget.Sublist	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	Updates a field ID that is to have unique values across the rows in the sublist.
Property	Sublist.displayType	string	The display style for a sublist.

Member Type	Name	Return Type / Value Type	Description
	Sublist.helpText	string	The inline help text for a sublist.
	Sublist.label	string	The label for a sublist.
	Sublist.lineCount	number (read-only)	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	Description
Property	Tab.helpText	string	The inline help text for a tab or subtab.
	Tab.label	string	The label for a tab or subtab.

N/ui/serverWidget Module Script Sample

The following code 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: 'textfield',
                    type: serverWidget.FieldType.TEXT,
                    label: 'Text'
                });
                field.layoutType = serverWidget.FieldLayoutType.NORMAL;
                field.breakType = serverWidget.FieldBreakType.STARTCOL;
                form.addField({
                    id: 'datefield',
                    type: serverWidget.FieldType.DATE,
                    label: 'Date'
                });
                form.addField({
                    id: 'currencyfield',
                    type: serverWidget.FieldType.CURRENCY,
                    label: 'Currency'
                });
                var select = form.addField({
                    id: '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 + ' ' + sublistFi
eld2);
}
}

return {
    onRequest: onRequest
};
});

```

serverWidget.Assistant

Object Description	Encapsulates a scriptable, multi-step NetSuite assistant. Each page of the assistant is defined by a step. All data and states for an assistant are tracked automatically throughout the user's session until completion of the assistant. You can create a new assistant with the serverWidget.createAssistant(options) method. After you create an <code>Assistant</code> object, you can: <ul style="list-style-type: none">■ Build and run an assistant in your NetSuite account.■ Add a variety of scriptable elements to the assistant including fields, steps, buttons, tabs, and sublists. For a complete list of this object's methods and properties, see Assistant Object Members .
Supported Script Types	Suitelets
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
//Add additional code
```

Assistant.addField(options)

Method Description	Adds a field to an assistant.
Returns	serverWidget.Field object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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.	Version 2015 Release 2
options.label	string	required	The label for this field.	Version 2015 Release 2
options.type	string	required	The field type. Use the serverWidget.FieldType enum to set this value.	Version 2015 Release 2
<p>Note: 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>				
options.source	string	optional	The internalId or scriptId of the source list for this field. Use the serverWidget.FieldType enum to set this value.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<p>Note: If you want to add custom options on a select field, you must set the source parameter to NULL.</p> <p>Important: 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>	
<code>options.container</code>	string	optional	The internal ID of the field group to place this field in.	Version 2016 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/ui/serverWidget Module Script Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addField({
    id : 'idname',
    type : serverWidget.FieldType.TEXT,
    label : 'Sample label'
});
...
//Add additional code
```

Assistant.addFieldGroup(options)

Method Description	Adds a field group to the assistant. By default, the field group is collapsible and appears expanded on the assistant page. To change this behavior, set the <code>FieldGroup.isCollapsed</code> and <code>FieldGroup.isCollapsible</code> properties.
---------------------------	--

Returns	serverWidget.FieldGroup object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	The internal ID for the field group.	Version 2015 Release 2
options.label	string	required	The label for the field group.	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addFieldGroup({
    id : 'idname',
    label : 'Sample label'
});
...
//Add additional code
```

Assistant.addStep(options)

Method Description	Adds a step to an assistant. If you want to create help text for the step, you can use AssistantStep.helpText on the object returned.
Returns	serverWidget.AssistantStep object
Supported Script Types	Suitelets

Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 step (for example, 'entercontacts').	Version 2015 Release 2
options.label	string	required	The label for this step (for example, 'Enter Contacts'). By default, the step appears vertically in the left panel of the assistant.	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
...
//Add additional code
```

Assistant.addSublist(options)

Method Description	Adds a sublist to an assistant.
	 Note: Only inline editor sublists are added. Other sublist types are not supported.
Returns	serverWidget.Sublist object
Supported Script Types	Suitelets

Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	The internal ID for the sublist.	Version 2015 Release 2
options.label	string	required	The label for the sublist.	Version 2015 Release 2
options.type	string	required	The type of sublist to add. For more information about possible values, see serverWidget.SublistType .	Version 2016 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/ui/serverWidget Module Script Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addSublist({
    id : 'idname',
    label : 'Sample label',
    type : serverWidget.SublistType.LIST
});
...
//Add additional code
```

Assistant.getField(options)

Method Description	Returns a field object on an assistant page.
Returns	serverWidget.Field
Supported Script Types	Suitelets

Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addField({
    id : 'idname',
    type : serverWidget.FieldType.TEXT,
    label : 'Sample label'
});
var field = assistant.getField({
    id: 'idname'
});
...
//Add additional code
```

Assistant.getFieldGroup(options)

Method Description	Returns a field group on an assistant page.
Returns	<code>serverWidget.FieldGroup</code> object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//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()

Method Description	Retrieves all the internal IDs for field groups in an assistant.
Returns	string[]
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//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()

Method Description	Gets all the internal IDs for fields in an assistant.
Returns	string[]
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```

//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addField({
    id : 'idname',
    label : 'Sample label'
});
var fieldid = assistant.getFieldIds();
...
//Add additional code

```

Assistant.getFieldIdsByFieldGroup(fieldGroup)

Method Description	Gets all field IDs in the assistant field group.
Returns	string[]
Supported Script Types	Suitelets

Governance	None
Module	N/ui/serverWidget Module
Since	Version 2016 Release 1

Parameters

Parameter	Type	Required / Optional	Description	Since
fieldGroup	string	required	The internal ID of the field group.	Version 2016 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/ui/serverWidget Module Script Sample](#).

```
//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()

Method Description	Gets the last action taken by the user. To identify the step that the last action came from, use Assistant.getLastStep() .
Returns	string
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module

Since	Version 2015 Release 2
-------	------------------------

Syntax



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 Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
if (assistant.getLastAction() == serverWidget.AssistantSubmitAction.CANCEL) {
    ...
}
...
//Add additional code
```

Assistant.getLastStep()

Method Description	Gets the step that the last submitted action came from.
Returns	A <code>serverWidget.AssistantStep</code> object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
var lastStep = assistant.getLastStep();
...
//Add additional code
```

Assistant.getNextStep()

Method Description	Gets the next step corresponding to the user's last submitted action in the assistant. If you need information about the last step, use Assistant.getLastStep() before you use this method.
Returns	<code>serverWidget.AssistantStep</code> object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
var nextStep = assistant.getNextStep();
...
//Add additional code
```

Assistant.getStep(options)

Method Description	Returns a step in an assistant.
Returns	<code>serverWidget.AssistantStep</code> object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 step.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//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()

Method Description	Gets the total number of steps in an assistant.
Returns	The total count of assistant steps as a number
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//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()

Method Description	Gets all the steps in an assistant.
Returns	serverWidget.AssistantStep[] object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```

//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)

Method Description	Returns a sublist in an assistant.
Returns	serverWidget.Sublist object

Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//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()

Method Description	Gets the IDs for all the sublists in an assistant.
Returns	string[]
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module

Since

Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addSublist({
    id : 'idname',
    label : 'Sample label',
    type : serverWidget.SublistType.LIST
});
var sublistid = assistant.getSublistIds();
...
//Add additional code
```

Assistant.hasErrorHtml()

Method Description	Determine whether an assistant has an error message to display for the current step.
Returns	boolean true if Assistant.errorHtml contains a value.
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
if (assistant.hasErrorHtml()) {
```

```

    ...
}

...
//Add additional code

```

Assistant.isFinished()

Method Description	Indicates whether all steps in an assistant are completed. If set to <code>true</code> , the assistant is finished and a completion message displays. To set the text for the completion message, use the Assistant.finishedHtml property.
Returns	<code>boolean true false</code>
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```

//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
if (assistant.isFinished()) {
    ...
}
...

//Add additional code

```

Assistant.sendRedirect(options)

Method Description	Manages redirects in an assistant. This method also addresses the case in which one assistant redirects to another assistant. In this scenario, the second assistant must return to the first assistant if the user Cancels or Finishes. This method, when used in the second assistant, ensures that users are redirected back to the first assistant.
Returns	<code>void</code>

Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//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)

Method Description	Defines a splash message.
Returns	void
Supported Script Types	Suitelets
Governance	None

Module	N/ui/serverWidget Module
Since	Version 2015 Release 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.	Version 2015 Release 2
options.text1	string	required	Text for the splash screen	Version 2015 Release 2
options.text2	string	optional	Text for a second column on the splash screen, if desired.	Version 2015 Release 2

Syntax



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 Sample](#).

```
...
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)

Method Description	Sets the default values of an array of fields that are specific to the assistant.
Returns	void
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module

Since	Version 2016 Release 1
-------	------------------------

Parameters

Parameter	Type	Required / Optional	Description	Since
values	object[]	required	An array of fields to update.	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addField({
    id : 'idname',
    type : serverWidget.FieldType.TEXT,
    label : 'Sample label'
});
assistant.updateDefaultValues({
    idname : "New Default Value"
});
...
//Add additional code
```

Assistant.clientScriptFileId

Property Description	The file cabinet ID of client script file to be used in this assistant.
Type	number
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 Assistant.clientScriptModulePath .

Syntax



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 Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.clientScriptId = 32;
...
//Add additional code
```

Assistant.clientScriptModulePath

Property Description	The relative path to the client script file to be used in this assistant.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2016 Release 2

Errors

Error Code	Thrown If
PROPERTY_VALUE_CONFLICT	You attempted to set this value when the Assistant.clientScriptFileId property value has already been specified. For more information, see Assistant.clientScriptFileId .

Syntax



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 Sample](#).

```
//Add additional code
...
objAssistant.clientScriptModulePath = 'SuiteScripts/assistantBehavior.js';
...
//Add additional code
```

Assistant.currentStep

Property Description	Identifies the current step.
----------------------	------------------------------

	You can set any step as the current step.
Type	serverWidget.AssistantStep (read-only)
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

⚠ 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 Sample](#).

```
//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

Property Description	Error message text for the current step. Optionally, you can use HTML tags to format the message.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

⚠ 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 Sample](#).

```
//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

Property Description	The text to display after the assistant finishes. For example "You have completed the Small Business Setup Assistant. Take the rest of the day off". To trigger display of the completion message, call Assistant.isFinished() .
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

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 Sample](#).

```
//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

Property Description	Indicates whether to show or hide the Add to Shortcuts link that appears in the top-right corner of an assistant page. By default, the value is <code>false</code> – the Add to Shortcuts link is visible in the UI. If set to <code>true</code> , the Add To Shortcuts link is not visible on an Assistant page.
Type	boolean <code>true</code> <code>false</code>
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

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 Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
```

```
assistant.hideAddToShortcutsLink = true;
...
//Add additional code
```

Assistant.hideStepNumber

Property Description	Indicates whether assistant steps are displayed with numbers. By default, the value is <code>false</code> , which means that steps are numbered. If set to <code>true</code> , the assistant does not use step numbers. To change step ordering, set Assistant.isNotOrdered .
Type	<code>boolean true false</code>
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//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

Property Description	Indicates whether steps must be completed in a particular sequence. If steps are ordered, users must complete the current step before proceeding to the next step. The default value is <code>false</code> – steps are ordered. Ordered steps appear vertically in the left panel of the assistant. If set to <code>true</code> , steps can be completed in any order. In the UI, unordered steps appear horizontally and below the assistant title
Type	<code>boolean true false</code>
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//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

Property Description	The title for the assistant. The title appears at the top of all assistant pages. This value overrides the title specified in serverWidget.createAssistant(options) .
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
var title = assistant.title;
...
//Add additional code
```

serverWidget.AssistantStep

Object Description	Encapsulates a step within a custom NetSuite assistant. Create a step by calling Assistant.addStep(options) . For a complete list of this object's methods and properties, see AssistantStep Object Members .
Supported Script Types	Suitelets
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
var assistantStep = assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
...
//Add additional code
```

AssistantStep.getFieldIds()

Method Description	Gets the IDs for all the fields in a step.
Returns	string[]
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//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)

Method Description	Gets the IDs for all the sublist fields (line items) in a step.
Returns	string[]
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.group	string	required	The sublist internal ID.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```

//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
var assistantStep = assistant.addStep({
    id : 'idname',

```

```

        label : 'Sample label'
    });
var sublist = assistant.addSublist({
    id : 'sublistid',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Editor'
});
var sublistFieldIds = assistantStep.getSublistFieldIds({
    group : 'sublistid'
});
...
//Add additional code

```

AssistantStep.getLineCount(options)

Method Description	Gets the number of lines on a sublist in a step.
	<p>Note: The first line number on a sublist is 0 (not 1).</p>
Returns	The count of line items on a sublist as a number
	<p>Note: if the sublist does not exist, -1 is returned.</p>
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.group	string	required	The sublist internal ID.	Version 2015 Release 2

Syntax

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 Sample](#).

```

//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()

Method Description	Gets the IDs for all the sublists submitted in a step.
Returns	string[]
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```

//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
var assistantStep = assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
var sublist = assistant.addSublist({
    id : 'sublistid',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Editor'
});
var submittedSublistId = assistantStep.getSubmittedSublistIds();
...
//Add additional code

```

AssistantStep.getSublistValue(options)

Method Description	Gets the current value of a sublist field (line item) in a step.
Returns	The value of a sublist field as a string
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.group	string	required	The internal ID of the sublist.	Version 2015 Release 2
options.id	string	required	The internal ID of the sublist field.	Version 2015 Release 2
options.line	number	required	The line number for the sublist field.  Note: The first line number on a sublist is 0 (not 1).	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//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)

Method Description	Gets the current value(s) of a field or mult-select field.
Returns	string[]
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	The internal ID of a field.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```

//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

Property Description	The help text for a step.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

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 Sample](#).

```

//Add additional code ...
assistantStep.helpText = 'Help Text Goes Here.';
...
//Add additional code

```

AssistantStep.id

Property Description	The internal ID of the step.
Type	string (read-only)
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

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 Sample](#).

```

//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

Property Description	The label for the step.
	 Note: To create a label when the step is first added to the assistant, you can use the Assistant.addStep(options) method.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

 **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 Sample](#).

```

//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

Property Description	Indicates where this step appears sequentially in the assistant.
Type	The index of this step as a number.
	 Note: A sequence of assistant steps starts at 1.
Module	N/ui/serverWidget Module

Since	Version 2015 Release 2
-------	------------------------

Syntax



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 Sample](#).

```
//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

Object Description	Encapsulates button that appears in a UI object. To add a button, use Form.addButton(options) or Sublist.addButton(options) . 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.
---------------------------	--



Note: Currently you cannot use SuiteScript to add or remove a custom button to or from the More Actions menu. You can, however, do this using SuiteBuilder point-and-click customization. See the help topic [Configuring Buttons and Actions](#).

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

Supported Script Types	Suitelets
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
```

```

...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var button = form.addButton({
    id : 'buttonid',
    label : 'Test'
});
...
//Add additional code

```

Button.isDisabled

Property Description	Indicates whether a button is grayed-out and disabled. The default value is <code>false</code> . If set to true, the button appears grayed-out in the and cannot be clicked.
Type	<code>boolean true false</code>
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```

//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

Property Description	Indicates whether the button is hidden in the UI. The default value is <code>false</code> – the button is visible. If set to true, the button is not visible in the UI.
-----------------------------	---

	 Note: This property is supported on custom buttons and on some standard NetSuite buttons. For a list of supported standard buttons, see the help topic Button IDs .
Type	boolean true false
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//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

Property Description	The label for the button. 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.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//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

Object Description	Encapsulates a NetSuite field. To add a Field object, use Assistant.addField(options) , Form.addField(options) , or Sublist.addField(options) . For a complete list of this object's methods and properties, see Field Object Members .
Supported Script Types	Suitelets
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```

//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
...
//Add additional code

```

Field.addSelectOption(options)

Method Description	Adds the select options that appears in the dropdown of a field.
---------------------------	--

	 Important: After you create a select or multi-select field that is sourced from a record or list, you cannot add additional values with <code>Field.addSelectOption(options)</code> . The select values are determined by the source record or list.
Returns	<code>void</code>
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
<code>options.value</code>	<code>string</code>	required	The internal ID of this select option.	Version 2015 Release 2
<code>options.text</code>	<code>string</code>	required	The label for this select option.	Version 2015 Release 2
<code>options.isSelected</code>	<code>boolean true 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> .	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var selectField = form.addField({
    id : 'selectfield',
    type : serverWidget.FieldType.SELECT,
    label : 'Select'
});
selectField.addSelectOption({
    value : 'a',
    text : 'Albert'
});
```

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

Field.getSelectOptions(options)

Method Description	Obtains a list of available options on a select field. The internal ID and label of the options for a select field as name/value pairs is returned. The first 1,000 available options are returned. If you attempt to get select options on a field that is not a select field, or if you reference a field that does not exist on the form, null is returned.
Returns	Object[]
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

i Note: 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> <th>Since</th> </tr> </thead> <tbody> <tr> <td>options.filter</td> <td>string</td> <td>optional</td> <td>A search string to filter the select options that are returned. For example, if there are 50 select options available, and 10 of the options contains 'John', e.g. "John Smith" or "Shauna Johnson", only those 10 options will be returned. Filter values are case insensitive. The filters 'John' and 'john' will return the same select options.</td> <td>Version 2015 Release 2</td> </tr> <tr> <td>options.filteroperator</td> <td>string</td> <td>optional</td> <td>Supported operators are <code>contains</code> <code>is</code> <code>startswith</code>. If not specified, defaults to the <code>contains</code> operator.</td> <td>Version 2015 Release 2</td> </tr> </tbody> </table>	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. "John Smith" or "Shauna Johnson", only those 10 options will be returned. Filter values are case insensitive. The filters 'John' and 'john' will return the same select options.	Version 2015 Release 2	options.filteroperator	string	optional	Supported operators are <code>contains</code> <code>is</code> <code>startswith</code> . If not specified, defaults to the <code>contains</code> operator.	Version 2015 Release 2
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. "John Smith" or "Shauna Johnson", only those 10 options will be returned. Filter values are case insensitive. The filters 'John' and 'john' will return the same select options.	Version 2015 Release 2											
options.filteroperator	string	optional	Supported operators are <code>contains</code> <code>is</code> <code>startswith</code> . If not specified, defaults to the <code>contains</code> operator.	Version 2015 Release 2											

Syntax

⚠ 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 Sample](#).

```
//Add additional code
```

```

...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var selectField = form.addField({
    id : '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)

Method Description	Sets the help text for the field. When the field label is clicked, a popup displays the help text defined using this method.
Returns	The serverWidget.Field object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.help	string	required	The text in the field help popup.	Version 2015 Release 2
options.showInlineForAssistant	boolean true false	optional	If set to <code>true</code> , the field help will display inline below the field on the assistant, and in a field help popup. The default value is <code>false</code> — the field help appears in a popup when the field label is clicked and does not appear inline.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<p>Note: The <code>inline</code> parameter is available only to <code>serverWidget.Field</code> objects that have been added to <code>serverWidget.createAssistant(options)</code> objects.</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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.setHelpText({
    help : "Help Text Goes Here."
});
...
//Add additional code
```

Field.updateBreakType(options)

Method Description	Updates the break type used to add a break in flow layout for the field.
Returns	<code>serverWidget.Field</code> object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2016 Release 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.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateBreakType({
    breakType : serverWidget.FieldBreakType.STARTCOL
});
...
//Add additional code
```

Field.updateDisplaySize(options)

Method Description	Updates the width and height of the field. Only supported on multi-selects, long text, rich text, and fields that get rendered as INPUT (type=text) fields. This function is not supported on list/record fields.
Returns	serverWidget.Field object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2016 Release 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.	Version 2015 Release 2
options.width	number	required	The new width of the field.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateDisplaySize({
    height : 60,
    width : 100
});
...
//Add additional code
```

Field.updateDisplayType(options)

Method Description	Updates the display type for the field.
Returns	serverWidget.Field object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2016 Release 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 serverWidget.FieldDisplayType .	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateDisplayType({
    displayType : serverWidget.FieldDisplayType.HIDDEN
});
...
//Add additional code
```

Field.updateLayoutType(options)

Method Description	Updates the layout type for the field.
Returns	serverWidget.Field object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2016 Release 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.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateLayoutType({
    layoutType : serverWidget.FieldLayoutType.NORMAL
});
...
//Add additional code
```

Field.alias

Property Description	An alternate name that you can assign to a <code>serverWidget.Field</code> object. By default, the alias is equal to the field's internal ID. This property is only supported on scripted fields created using the N/ui/serverWidget Module .
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

 **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 Sample](#).

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

```

var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.alias = 'fieldId';
...
//Add additional code

```

Field.defaultValue

Property Description	The default value for this field. If you pass an empty string, the field defaults to a blank field in the UI. This property is supported only on scripted fields created using the N/ui/serverWidget Module .
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```

//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.defaultValue = 'Insert Text Here.';
...
//Add additional code

```

Field.id

Property Description	The field internal ID.
Type	string (read-only)
Module	N/ui/serverWidget Module

Since	Version 2015 Release 2
-------	------------------------

Syntax

⚠ 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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
var fieldId = field.id;
...
//Add additional code
```

Field.isMandatory

Property Description	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 N/ui/serverWidget Module .
Type	<code>boolean true false</code>
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

⚠ 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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
```

```
field.isMandatory = true;
...
//Add additional code
```

Field.label

Property Description	The field label. There is a 40-character limit for custom field labels.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

⚠ 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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
var label = field.label;
...
//Add additional code
```

Field.linkLabel

Property Description	The text displayed for a link in place of the URL.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

⚠ 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 Sample](#).

```
//Add additional code
```

```

...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.URL,
    label : 'URL'
});
field.linkText = 'NetSuite';
...
//Add additional code

```

Field.maxLength

Property Description	The maximum length, in characters, of the field (only valid for text, rich text, long text, and textarea fields). This property is supported only on scripted fields created using the N/ui/serverWidget Module .
Type	number
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```

//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.maxLength = 64;
...
//Add additional code

```

Field.padding

Property Description	The number of empty vertical character spaces above the field. This property is supported only on scripted fields created using the N/ui/serverWidget Module .
-----------------------------	---

Type	number
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

⚠ 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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.padding = 50;
...
//Add additional code
```

Field.richTextHeight

Property Description	The height of a rich text field, in pixels. The minimum value is 100 pixels and the maximum value is 500 pixels.
	ℹ Note: Rich Text Editing must be enabled.
Type	number
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

⚠ 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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
```

```

var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.RICHTEXT,
    label : 'Rich Text'
});
field.richTextHeight = 50;
...
//Add additional code

```

Field.richTextWidth

Property Description	The width of a rich text field, in pixels. The minimum value is 250 pixels and the maximum value is 800 pixels.
	Note: Rich Text Editing must be enabled.
Type	number
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

⚠ 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 Sample](#).

```

//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.RICHTEXT,
    label : 'Rich Text'
});
field.richTextWidth = 100;
...
//Add additional code

```

Field.type

Property Description	The field type. For example, text, date, currency, select, checkbox etc.
Type	string (read-only)
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
var fieldtype = field.type;
...
//Add additional code
```

serverWidget.FieldGroup

Object Description	Encapsulates a field group on <code>serverWidget.createAssistant(options)</code> objects and on <code>serverWidget.Form</code> objects. For a complete list of this object's properties, see FieldGroup Object Members .
Supported Script Types	Suitelets
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var fieldgroup = form.addFieldGroup({
    id : 'fieldgroupid',
    label : 'Field Group'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
```

```

        label : 'Text',
        container : 'fieldgroupid'
    });
...
//Add additional code

```

FieldGroup.isBorderHidden

Property Description	Indicates whether the field group can be collapsed. The default value is <code>false</code> - the field group border is not hidden. If set to true, a border around the field group is displayed.
Type	<code>boolean true false</code>
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```

//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var fieldgroup = form.addFieldGroup({
    id : 'fieldgroupid',
    label : 'Field Group'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    container : 'fieldgroupid'
});
fieldgroup.isBorderHidden = true;
...
//Add additional code

```

FieldGroup.isCollapsible

Property Description	Indicates whether the field group can be collapsed. The default value is <code>false</code> - 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 <code>serverWidget.createAssistant(options)</code> objects
-----------------------------	--

Type	boolean true false
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var fieldgroup = form.addFieldGroup({
    id : 'fieldgroupid',
    label : 'Field Group'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    container : 'fieldgroupid'
});
fieldgroup.isCollapsible = true;
...
//Add additional code
```

FieldGroup.isCollapsed

Property Description	Indicates whether field group is collapsed or expanded. The default value is <code>false</code> – when the page loads, the field group will not appear collapsed. If set to <code>true</code> , the field group is collapsed. Only supported for fields on <code>serverWidget.createAssistant(options)</code> objects
Type	boolean true false
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
```

```

...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var fieldgroup = form.addFieldGroup({
    id : 'fieldgroupid',
    label : 'Field Group'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    container : 'fieldgroupid'
});
var collapsed = fieldgroup.isCollapsed;
...
//Add additional code

```

FieldGroup.isSingleColumn

Property Description	Indicates whether the field group is aligned. The default value is <code>false</code> .
Type	<code>boolean true false</code>
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```

//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var fieldgroup = form.addFieldGroup({
    id : 'fieldgroupid',
    label : 'Field Group'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    container : 'fieldgroupid'
});
var aligned = fieldgroup.isSingleColumn;
...

```

```
//Add additional code
```

FieldGroup.label

Property Description	The label for the field group.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var fieldgroup = form.addFieldGroup({
    id : 'fieldgroupid',
    label : 'Field Group'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    container : 'fieldgroupid'
});
var label = fieldgroup.label;
...
//Add additional code
```

serverWidget.Form

Object Description	Encapsulates a NetSuite-looking form After you create a <code>Form</code> object, you can: <ul style="list-style-type: none"> ■ Add a variety of scriptable elements to the form including fields, links, buttons, tabs, and sublists. For a complete list of this object's methods and properties, see Form Object Members .
Supported Script Types	Suitelets
Module	N/ui/serverWidget Module

Since	Version 2015 Release 2
-------	------------------------

Syntax



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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
...
//Add additional code
```

Form.addButton(options)

Method Description	Adds a button to a form.
Returns	serverWidget.Button object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 <code>custpage</code> . For example, if you add a button that appears as Update Order , the button internal ID should be something similar to <code>custpage_updateorder</code> .	Version 2015 Release 2
options.label	string	required	The label for this button.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.functionName	string	optional	The function name to be triggered on a click event.	Version 2016 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/ui/serverWidget Module Script Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});

form.addButton({
    id : 'buttonid',
    label : 'Test'
});
...
//Add additional code
```

Form.addCredentialField(options)

Method Description	Adds a field that lets you store credentials in NetSuite to be used when invoking services provided by third parties. The GUID generated by this field can be reused multiple times until the script executes again. For example, when executing credit card transactions, merchants need to store credentials in NetSuite that are used to communicate with Payment Gateway providers. Note the following about this method: <ul style="list-style-type: none"> ■ Credentials associated with this field are stored in encrypted form. ■ No piece of SuiteScript holds a credential in clear text mode. ■ NetSuite reports or forms will never provide to the end user the clear text form of a credential. ■ Any exchange of the clear text version of a credential with a third party must occur over SSL. ■ For no reason will NetSuite ever log the clear text value of a credential (for example, errors, debug message, alerts, system notes, and so on).
Returns	serverWidget.Field object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 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.	Version 2015 Release 2
options.label	string	required	The label for the credential field.	Version 2015 Release 2
options.restrictToDomains	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.	Version 2015 Release 2
options.restrictToScriptIds	string string[]	required	The IDs of the scripts that are allowed to use this credential field. For example, 'customscript_my_script'.	Version 2015 Release 2
options.restrictToCurrentUser	boolean true false	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> – 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.	Version 2015 Release 2
options.container	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.	Version 2016 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/ui/serverWidget Module Script Sample](#).

```
//Add additional code
```

```

...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addCredentialField({
    id : 'username',
    label : 'Username',
    restrictToDomains : 'www.mysite.com',
    restrictToScriptIds : 'customscript_my_script',
    restrictToCurrentUser : false,
});
...
//Add additional code

```

Form.addField(options)

Method Description	Adds a field to a form.
Returns	serverWidget.Field object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 Purchase Details , the field internal ID should be something similar to <code>custpage_purchasedetails</code> or <code>custpage_purchase_details</code> .	Version 2015 Release 2
options.label	string	required	The label for this field.	Version 2015 Release 2
options.type	string	required	The field type for the field. Use the serverWidget.FieldType enum to define the field type.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.source	string	optional	<p>The internalId or scriptId of the source list for this field if it is a select (List/Record) or multi-select field.</p> <p>Important: 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>Note: For radio fields only, the <code>source</code> parameter must contain the internal ID for the field.</p> <p>For more information about working with radio buttons, see the help topic Working with Radio Buttons.</p>	Version 2015 Release 2
options.container	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.	Version 2016 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/ui/serverWidget Module Script Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
...
//Add additional code
```

Form.addFieldGroup(options)

Method Description	Adds a group of fields to a form.
Returns	<code>serverWidget.FieldGroup</code> object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

 **Note:** 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.	Version 2015 Release 2
<code>options.label</code>	string	required	The label for this field group.	Version 2015 Release 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.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var fieldgroup = form.addFieldGroup({
    id : 'fieldgroupid',
    label : 'Field Group'
});
...
//Add additional code
```

Form.addPageLink(options)

Method Description	Adds a link to a form.
---------------------------	------------------------

	 Note: You cannot choose where the page link appears.
Returns	void
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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.	Version 2015 Release 2
options.type	string	required	The type of page link to add. Use the <code>serverWidget.FormPageLinkType</code> enum to set the value.	Version 2015 Release 2
options.url	string	required	The URL for the link.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//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)

Method Description	Adds a reset button to a form. The reset buttons allows a user to clear the entries.
---------------------------	--

Returns	serverWidget.Button object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.label	string	optional	The label used for this button. If no label is provided, the label defaults to Reset .	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addResetButton({
    label : 'Reset Button'
});
...
//Add additional code
```

Form.addSecretKeyField(options)

Method Description	Adds a secret key field to the form. This key can be used in crypto modules to perform encryption or hashing.
Returns	serverWidget.Field object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module

Since	Version 2016 Release 1
-------	------------------------

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.id	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.	Version 2016 Release 1
options.restrictToCurrentUser	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> - multiple users can use the key. If set to <code>true</code> , the secret key applies to a single user.	Version 2016 Release 1
options.restrictToScriptIds	string or string[]	optional	The script ID of the script that is allowed to use this field.	Version 2016 Release 1
options.container	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.	Version 2016 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/ui/serverWidget Module Script Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addSecretKeyField({
```

```

    id : 'password',
    restrictToScriptIds : 'customscript_my_script',
    restrictToCurrentUser : false,
});
...
//Add additional code

```

Form.addSublist(options)

Method Description	Add a sublist to a form.
	<p>Note: If the row count exceeds 25, sorting is not supported on static sublists created using this method.</p>
Returns	A serverWidget.Sublist object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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. 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 Purchase Details , the sublist internal ID should be something equivalent to <code>custpage_purchasedetails</code> or <code>custpage_purchase_details</code> .	Version 2015 Release 2
options.label	string	required	The label for this sublist.	Version 2015 Release 2
options.tab	string	optional	The tab under which to display this sublist. If empty, the sublist is added to the main tab.	Version 2015 Release 2
options.type	string	required	The sublist type. Use the serverWidget.SublistType enum to set the value.	Version 2015 Release 2

Syntax

⚠ 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 Sample](#).

```
//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)

Method Description	Adds a submit button to a form.
	i Note: If the row count exceeds 25, sorting is not supported on static sublists created using this method.
Returns	serverWidget.Button object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2016 Release 1

Parameters

i 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".	Version 2016 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/ui/serverWidget Module Script Sample](#).

```
//Add additional code
```

```

...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addSubmitButton({
    label : 'Submit Button'
});
...
//Add additional code

```

Form.addSubtab(options)

Method Description	Adds a subtab to a form.
	 Important: If you add only one subtab, the label you define for the subtab will not appear in the UI. You must define two subtabs for subtab labels to appear.
Returns	serverWidget.Tab object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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. 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 Purchase Details , the subtab internal ID should be something similar to <code>custpage_purchasedetails</code> or <code>custpage_purchase_details</code> .	Version 2015 Release 2
options.label	string	required	The label for this subtab.	Version 2015 Release 2
options.tab	string	optional	The tab under which to display this sublist. If empty, the sublist is added to the main tab.	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addSubtab({
    id : 'subtabId',
    label : 'Subtab'
});
...
//Add additional code
```

Form.addTab(options)

Method Description	Adds a tab to a form.
Returns	serverWidget.Tab object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 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 you add a subtab that appears as Purchase Details , the subtab internal ID should be something similar to <code>custpage_purchasedetails</code> or <code>custpage_purchase_details</code> .	Version 2015 Release 2
options.label	string	required	The label for this tab.	Version 2015 Release 2

Syntax

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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var tab = form.addTab({
    id : 'tabId',
    label : 'Tab'
});
...
//Add additional code
```

Form.getButton(options)

Method Description	Returns a Button object by internal ID.
Returns	serverWidget.Button object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 button. Internal IDs must be in lowercase and contain no spaces.	Version 2015 Release 2

Syntax

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 Sample](#).

```
//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)

Method Description	Returns a Field object by internal ID.
Returns	serverWidget.Field object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 field. Internal IDs must be in lowercase and contain no spaces.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```

//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});

```

```

var field = form.getField({
    id : 'textfield'
});
...
//Add additional code

```

Form.getSublist(options)

Method Description	Returns a Sublist object by internal ID.
Returns	serverWidget.Sublist object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```

//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)

Method Description	Returns a subtab by internal ID.
Returns	<code>serverWidget.Tab</code> object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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. Internal IDs must be in lowercase and contain no spaces.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addTab({
    id : 'subtabId',
    label : 'Subtab'
});
var subtab = form.getSubtab({
    id : 'subtabId'
});
...
//Add additional code
```

Form.getTab(options)

Method Description	Returns a tab object from its internal ID.
---------------------------	--

Returns	serverWidget.Tab object.
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2016 Release 1

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 tab to retrieve.	Version 2016 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/ui/serverWidget Module Script Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addTab({
    id : 'tabId',
    label : 'Tab'
});
var tab = form.getTab({
    id : 'tabId'
});
...
//Add additional code
```

Form.getTabs()

Method Description	Returns an array that contains all the tabs in a form.
Returns	serverWidget.Tab[] objects
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

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

Syntax

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

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addTab({
    id : 'tabId',
    label : 'Tab'
});
var tab = form.getTabs();
...
//Add additional code
```

Form.insertField(options)

Method Description	Inserts a field in front of another field.
Returns	void
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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.	Version 2016 Release 1
options.nextfield	string	required	The internal ID name of the field you are inserting a field in front of.	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field1 = form.addField({
    id : 'textfield1',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
var field2 = form.addField({
    id : 'textfield2',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
form.insertField({
    field : field2,
    nextfield : 'textfield1'
});
...
//Add additional code
```

Form.insertSublist(options)

Method Description	Inserts a sublist in front of another sublist.
Returns	void
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters



Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.sublist	serverWidget.Sublist	required	The Sublist object to insert.	Version 2016 Release 1

Parameter	Type	Required / Optional	Description	Since
options.nextsublist	string	required	The internal ID name of the sublist you are inserting a sublist in front of.	Version 2015 Release 2

Syntax



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 Sample](#).

```
//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)

Method Description	Inserts a subtab in front of another subtab.
Returns	void
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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.	Version 2016 Release 1
options.nextsubtab	string	required	The internal ID name of the subtab you are inserting a subtab in front of.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//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)

Method Description	Inserts a tab in front of another tab.
Returns	void
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module

Since	Version 2015 Release 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.	Version 2016 Release 1
options.nexttab	string	required	The internal ID name of the tab you are inserting a tab in front of.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//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)

Method Description	Removes a button. This method can be used on custom buttons and certain built-in NetSuite buttons. For more information about built-in NetSuite buttons, see the help topic Button IDs .
---------------------------	---

Returns	void
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 button to remove. The internal ID must be in lowercase and contain no spaces.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addButton({
    id : 'buttonid',
    label : 'Test'
});
form.removeButton({
    id : 'buttonid'
});
...
//Add additional code
```

Form.updateDefaultValues(options)

Method Description	Updates the default values of multiple fields on the form.
Returns	void

Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2016 Release 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.	Version 2016 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/ui/serverWidget Module Script Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
form.setDefaultValue({
    textfield : 'Text Goes Here'
})
...
//Add additional code
```

Form.clientScriptFileId

Property Description	The internal file ID of client script file to be used in this form. Use this property when attaching an ad-hoc client script to a server-side script.
Type	number
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 Form.clientScriptModulePath .

Syntax

 **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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.clientScriptFileId = 32;
...
//Add additional code
```

Form.clientScriptModulePath

Property Description	The relative path to the client script file to be used in this form. Use this property when attaching an ad-hoc client script to a server-side script.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2016 Release 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 Form.clientScriptFileId .

Syntax

 **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 Sample](#).

```
//Add additional code
...
objForm.clientScriptModulePath = 'SuiteScripts/formBehavior.js';
...
```

```
//Add additional code
```

Form.title

Property Description	The title used for the form.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

⚠️ 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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var title = form.title;
...
//Add additional code
```

serverWidget.List

Object Description	Encapsulates a list. For a complete list of this object's methods and properties, see List Object Members .
Supported Script Types	Suitelets
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

⚠️ 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 Sample](#).

```
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
};
...
//Add additional code
```

List.addButton(options)

Method Description	Adds a button to a list.
Returns	serverWidget.Button object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 <code>custpage</code> if you are adding the button to an existing page. For example, if you add a button that appears as Update Order , the button internal ID should be something similar to <code>custpage_updateorder</code> .	Version 2015 Release 2
options.label	string	required	The label for this button.	Version 2015 Release 2
options.functionName	string	optional	The function name to call when clicking on this button.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});
list.addButton({
    id : 'buttonId',
    label : 'Test'
});
...
...
```

```
//Add additional code
```

List.addColumn(options)

Method Description	Adds a column to a list.
Returns	<code>serverWidget.ListColumn</code> object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 this column. The internal ID must be in lowercase, contain no spaces.	Version 2015 Release 2
<code>options.label</code>	string	required	The label for this column.	Version 2015 Release 2
<code>options.type</code>	string	required	The field type for this column.  Note: CHECKBOX field types are not supported. For more information about possible values, see <code>serverWidget.FieldType</code> .	Version 2015 Release 2
<code>options.align</code>	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>	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//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.addColumn(options)

Method Description	Adds a column containing Edit or Edit/View links to a Suitelet or Portlet list. These Edit or Edit/View links appear to the left of a previously existing column.
Returns	serverWidget.ListColumn object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.column	string	required	■ The internal ID of the column to the left of which the Edit/View Column will be added.	Version 2015 Release 2
options.showHrefCol	boolean true false	optional	If set to true, the URL for the link is clickable.	Version 2015 Release 2
options.showView	boolean true false	optional	If true then an Edit/View column will be added. Otherwise only an Edit column will be added.	Version 2015 Release 2

Syntax

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 Sample .
--

```
//Add additional code
```

```

...
var list = serverWidget.createList({
    title : 'Simple List'
});
list.addEditColumn({
    id : 'column1',
    showView : true
});
...
//Add additional code

```

List.addPageLink(options)

Method Description	Adds a link to a list.
Returns	void
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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.	Version 2015 Release 2
options.type	string	required	The type of page link to add. For more information about possible values, see serverWidget.FormPageLinkType .	Version 2015 Release 2
options.url	string	required	The URL for the link.	Version 2015 Release 2

Syntax

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 Sample](#).

```

//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)

Method Description	Adds a single row to a list.
Returns	serverWidget.List
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.row	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.	Version 2015 Release 2

Syntax

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 Sample](#).

```

//Add additional code  

...  

var list = serverWidget.createList({  

    title : 'Simple List'  

});  

list.addRow({  

    row : { columnid1 : 'value1', columnid2 : 'value2' }  

});  

...
//Add additional code

```

List.addRows(options)

Method Description	Adds multiple rows to a list.
Returns	serverWidget.ListColumn
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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.	Version 2015 Release 2

Syntax

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 Sample](#).

```
//Add additional code
...
list.addRows({
    rows : [{columnid1 : 'value1', columnid2 : 'value2'},
             {columnid1 : 'value2', columnid2 : 'value3'}]
});
...
//Add additional code
```

List.clientScriptFileId

Property Description	The file cabinet ID of client script file to be used in this list.
Type	number
Module	N/ui/serverWidget Module
Since	Version 2016 Release 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 List.clientScriptModulePath .

Syntax



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 Sample](#).

```
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});
list.clientScriptFileDialog = 123;
...
//Add additional code
```

List.clientScriptModulePath

Property Description	The relative path to the client script file to be used in this list.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2016 Release 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 List.clientScriptFileDialog .

Syntax



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 Sample](#).

```
//Add additional code
...
objList.clientScriptModulePath = 'SuiteScripts/listBehavior.js';
...
//Add additional code
```

List.style

Property Description	Sets the display style for this list. For more information about possible values, see serverWidget.ListStyle .
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});
list.style = serverWidget.ListStyle.REPORT;
...
//Add additional code
```

List.title

Property Description	Sets the list title.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});
var title = list.title;
...
//Add additional code
```

serverWidget.ListColumn

Object Description	Encapsulates a list column For a complete list of this object's methods and properties, see ListColumn Object Members .
Supported Script Types	Suitelets
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

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 Sample](#).

```
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});
var listcolumn = list.addColumn({
    id : 'column1',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    align : serverWidget.LayoutJustification.RIGHT
});
...
//Add additional code
```

ListColumn.addParamToURL(options)

Method Description	Adds a URL parameter (optionally defined per row) to the list column's URL.
Returns	<code>serverWidget.ListColumn</code> object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2016 Release 1

Parameters

i Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
<code>options.param</code>	string	required	The name for the parameter.

Parameter	Type	Required / Optional	Description
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 Sample](#).

```
//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)

Method Description	Sets the base URL for the list column.
Returns	<code>serverWidget.ListColumn</code>
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2016 Release 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

Parameter	Type	Required / Optional	Description
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 Sample](#).

```
//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

Property Description	This list column label.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2016 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/ui/serverWidget Module Script Sample](#).

```
//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

Object Description	Encapsulates a sublist on a serverWidget.Form or an serverWidget.createAssistant(options) object. To add a sublist, use Assistant.addSublist(options) or Form.addSublist(options) .
<p>Note: This object is read-only except for instances created via the serverWidget module using Suitelets or beforeLoad user event scripts.</p>	
	For a complete list of this object's methods and properties, see Sublist Object Members .
Supported Script Types	Suitelets
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```

//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)

Method Description	Adds a button to a sublist.
Returns	serverWidget.Button

Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 button. The internal ID must be in lowercase and without spaces.
options.label	string	required	The label for the button.
options.functionName	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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
sublist.addButton({
    id : 'buttonId',
    label : 'Test'
});
...
//Add additional code
```

Sublist.addField(options)

Method Description	Adds a field to a sublist.
--------------------	----------------------------

Returns	serverWidget.Field object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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.	Version 2015 Release 2
options.label	string	required	The label for this field.	Version 2015 Release 2
options.type	string	required	The field type. Use the <code>serverWidget.FieldType</code> enum to set this value. Note that the <code>INLINEHTML</code> value is not supported with this method.	Version 2015 Release 2
 Note: 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. Use this parameter if you are adding a select (List/Record) type of field.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<p>Note: If you want to add custom options on a select field, you must set the source parameter to NULL.</p> <p>Important: 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 Sample](#).

```
//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()

Method Description	Adds a Mark All and an Unmark All button to a <code>LIST</code> type of sublist.
Returns	A <code>serverWidget.Button[]</code> object
Supported Script Types	Suitelets

Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//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.addMarkAllButtons();
...
//Add additional code
```

Sublist.addButton()

Method Description	Adds a Refresh button to a LIST type of sublist.
Returns	serverWidget.Button object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//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.addRefreshButton();
...
//Add additional code

```

Sublist.getField(options)

Method Description	Returns a Field object on a sublist.
Returns	serverWidget.Field
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2016 Release 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 help topic SuiteScript Supported Sublists .

Syntax

 **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 Sample](#).

```

//Add additional code
...
var itemField = form.getSublist({id: 'item'}).getField({id: 'item'});
...
//Add additional code

```

Sublist.getSublistValue(options)

Method Description	Gets a field value on a sublist.
Returns	void
Supported Script Types	Suitelets
Governance	None

Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

<p>Note: The options parameter is a JavaScript object.</p>			
Parameter	Type	Required / Optional	Description
options.id	string	required	<p>The sublist internal ID (for example, use addressbook as the ID for the Address sublist). For more information about supported sublists, internal IDs, and field IDs, see the help topic SuiteScript Supported Sublists.</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 Sample](#).

```
//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 sublistvalue = sublist.getSublistValue({
    id : 'sublist',
    line: 1
})
...
//Add additional code
```

Sublist.setSublistValue(options)

Method Description	Sets the value of a sublist field.
Returns	void

Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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.  Note: The first line number on a sublist is 0 (not 1).
options.value	string	required	The value for the field being set.

Syntax

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

```
//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.setSublistValue({
    id : 'sublist',
    line : 2,
    value : "Text"
})
...
//Add additional code
```

Sublist.updateTotallingFieldId(options)

Method Description	Updates the ID of a field designated as a totalling column, which is used to calculate and display a running total for the sublist.
---------------------------	---

Returns	serverWidget.Sublist object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 Sample](#).

```
//Add additional code
...
sublist.updateTotallingFieldId({
    id : 'fieldId'
})
...
//Add additional code
```

Sublist.updateUniqueId(options)

Method Description	Updates a field ID that is to have unique values across the rows in the sublist.  Note: This method is available on inlineeditor and editor sublists only.
Returns	serverWidget.Sublist object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
sublist.addField({
    id : 'fieldId',
    type : serverWidget.FieldType.DATE,
    label : 'Date'
});
sublist.updateUniqueFieldId({
    id : 'fieldId'
})
...
//Add additional code
```

Sublist.displayType

Property Description	The display style for a sublist. Use the serverWidget.SublistDisplayType enum to set this value.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//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

Property Description	The inline help text for a sublist.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//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

Property Description	The label for this sublist.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
var label = sublist.label;
...
//Add additional code
```

Sublist.lineCount

Property Description	The number of line items on a sublist.
	Note: The first line number on a sublist is 0 (not 1).
Type	number (read-only)
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax

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 Sample](#).

```
//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

Object Description	Encapsulates a tab or subtab on a serverWidget.Form object. You can add a new tab or subtab to a form using one of the following methods:
	<ul style="list-style-type: none"> ■ Form.addSubtab(options) ■ Form.addTab(options) ■ Form.insertSubtab(options) ■ Form.insertTab(options)
	For a complete list of this object's properties, see Tab Object Members .
Supported Script Types	Suitelets
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```

//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var tab = form.addTab({
    id : 'tabId',
    label : 'Tab'
});
...
//Add additional code

```

Tab.helpText

Property Description	The inline help text for a tab or subtab.
-----------------------------	---

Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//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

Property Description	The label for a tab or subtab.
Type	string
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Syntax



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 Sample](#).

```
//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)

Method Description	Creates an assistant object.
Returns	<code>serverWidget.Assistant</code> object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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 assistant. This title appears at the top of all assistant pages.	Version 2015 Release 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 assistant. If set to <code>true</code> , the header on the assistant is hidden from view.	Version 2015 Release 2

Syntax

 **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 Sample](#).

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
```

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

serverWidget.createForm(options)

Method Description	Creates a form object.
Returns	serverWidget.Form object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.title	string	required	The title of the form.	Version 2015 Release 2
options.hideNavBar	boolean true false	optional	Indicates whether to hide the navigation bar menu. By default, set to false. The header appears in the top-right corner on the form. If set to true, the header on the assistant is hidden from view.	Version 2015 Release 2

Syntax



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 Sample](#).

```
//Add additional code ...  
var form = serverWidget.createForm({  
    title : 'Simple Form'  
});  
...
```

```
//Add additional code
```

serverWidget.createList(options)

Method Description	Instantiates a standalone list.
Returns	serverWidget.List object
Supported Script Types	Suitelets
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.title	string	required	The title of the list.	Version 2016 Release 1
options.hideNavBar	boolean true false	optional	Indicates whether to hide the navigation bar menu. By default, set to false. The header appears in the top-right corner on the form. If set to true, the header on the assistant is hidden from view.	Version 2016 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/ui/serverWidget Module Script Sample](#).

```
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});
...
//Add additional code
```

serverWidget.AssistantSubmitAction

Enum Description	Holds the string values for submit actions performed by the user. This enum is used to set the value of the <code>Assistant.getLastAction()</code> . After a <code>finish</code> action is submitted, by default, the text "Congratulations! You have completed the <assistant title>" appears on the finish page. In a non-sequential process (steps are unordered), <code>jump</code> is used to move to the user's last action.
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Values

- BACK
- CANCEL
- FINISH
- JUMP
- NEXT

Syntax

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

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
if (assistant.getLastAction() == serverWidget.AssistantSubmitAction.CANCEL) {
    ...
}
...
//Add additional code
```

serverWidget.FieldBreakType

Enum Description	Enumeration that holds the string values for supported field break types. This enum is used to set the value of the <code>breakType</code> parameter when <code>Field.updateBreakType(options)</code> is called.
-------------------------	--

	<p>Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Values

- NONE
- STARTCOL
- STARTROW

Syntax

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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateBreakType({
    breakType : serverWidget.FieldBreakType.STARTCOL
});
...
//Add additional code
```

serverWidget.FieldDisplayType

Enum Description	Enumeration that holds the string values for supported field display types. This enum is used to set the value of the <code>displayType</code> parameter when <code>Field.updateDisplayType(options)</code> is called.
Module	N/ui/serverWidget Module

Since	Version 2015 Release 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 snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateDisplayType({
    displayType: serverWidget.FieldDisplayType.HIDDEN
});
...
//Add additional code
```

serverWidget.FieldLayoutType

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

Since	Version 2015 Release 2
-------	------------------------

Values

- ENDROW
- NORMAL
- MIDROW
- OUTSIDE
- OUTSIDEBEL
OW
- OUTSIDEABO
VE
- STARTROW

Syntax



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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateLayoutType({
    layoutType = serverWidget.FieldLayoutType.NORMAL;
});
...
//Add additional code
```

serverWidget.FieldType

Enum Description	Enumeration that holds the values for supported field types. This enum is used to set the value of the <code>type</code> parameter when <code>Form.addField(options)</code> is called. i Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Module	N/ui/serverWidget Module

Since	Version 2015 Release 2
-------	------------------------

Values

- CHECKBOX ▪ LONGTEXT
- CURRENCY ▪ MULTISELEC
T
- DATE ▪ PASSPORT
- DATETIMETZ ▪ PERCENT
- EMAIL ▪ PHONE
- FILE ▪ SELECT
- FLOAT ▪ RADIO
- HELP ▪ RICHTEXT
- INLINEHTML ▪ TEXT
- INTEGER ▪ TEXTAREA
- IMAGE ▪ TIMEOFDAY
- LABEL ▪ URL

Syntax



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 Sample](#).

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
...
//Add additional code
```

serverWidget.FormPageLinkType

Enum Description	Enumeration that holds the string values for supported page link types on a form. This enum is used to set the value of the <code>type</code> parameter when <code>Form.addPageLink(options)</code> is called.
------------------	--

	<p>Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Values

- BREADCRUMB
- CROSSLINK

Syntax

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 Sample](#).

```
//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

Enum Description	<p>Enumeration that holds the string values for supported justification layouts. This enum is used to set the value of the align parameter when <code>List.addColumn(options)</code> is called.</p> <p>Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Values

- CENTER
- LEFT
- RIGHT

Syntax

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

```
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});
list.addColumn({
    id : 'column1',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    align : serverWidget.LayoutJustification.RIGHT
});
...
//Add additional code
```

serverWidget.ListStyle

Enum Description	Enumeration that holds the string values for supported list styles. This enum is used to set the value of the List.style property.
	<p>i Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Values

- 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 Sample](#).

```
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});
list.style = serverWidget.ListStyle.REPORT;
...
//Add additional code
```

serverWidget.SublistDisplayType

Enum Description	Enumeration that holds the string values for supported sublist display types. This enum is used to set the value of the Sublist.displayType property.
	<p> Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Values

- HIDDEN
- NORMAL

Syntax



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

```
//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

Enum Description	Enumeration that holds the string values for valid sublist types. This enum is used to define the <code>type</code> parameter when Form.addSublist(options) is called
Module	N/ui/serverWidget Module
Since	Version 2015 Release 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"> 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. 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.
LIST	This type of sublist has a fixed number of lines. You can update an existing line, but you cannot add lines to it.
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 Sample](#).

```
//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

```

N/url Module

Use the url module to determine URL navigation paths within NetSuite and format URL strings.

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

N/url Module Members

Member Type	Name	Return Type / Value Type	Description
Method	url.format(options)	string	Converts (serializes) URL query parameters into a string.
	url.resolveDomain(options)	string	Returns a domain name for a NetSuite account.
	url.resolveRecord(options)	string	Returns an internal URL string to a NetSuite record.
	url.resolveScript(options)	string	Returns an external or internal URL string to a script.
	url.resolveTaskLink(options)	string	Returns an internal URL for a tasklink.
Enum	url.HostType	enum	An enum used to populate the hostType parameter of the url.resolveDomain(options) method.

N/url Module Script Sample

The following script samples show how to use the url module.

These samples use the `require` function, so that you can copy each script into the debugger and test it, after making any necessary edits. Remember that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see [SuiteScript 2.0 – Script Architecture](#) and [SuiteScript 2.0 Script Types and Entry Points](#).



Important: Some values in these samples are placeholders. Before using these samples, replace this values with valid ones from your NetSuite account. If you run a script with an invalid value, the system may throw an error.

The following example retrieves the relative URL of a record. With the internal ID value used in this example, the returned output would be `/app/accounting/transactions/salesord.nl?id=6&e=T&compid=`, followed by the NetSuite account ID.

```

/**
 *@NApiVersion 2.x
 */
require(['N/url'],
    function(url) {

```

```

        var output = url.resolveRecord({
            recordType: 'salesorder',
            recordId: 6,
            isEditMode: true
        });
    });
}

```

The following example shows how to get the domain for calling a RESTlet.

```

/**
 * @NApiVersion 2.x
 */
require(['N/url'],
    function(url) {
        function resolveDomainUrl() {
            var sCompId = 'MSTRWLF';
            var output = url.resolveDomain({
                hostType: url.HostType.RESTLET,
                accountId: sCompId
            });
        }
        resolveDomainUrl();
    }
);

```

url.format(options)

Method Description	Creates a serialized representation of an object containing query parameters. Use the returned value to build a URL query string.
Returns	URL as a string
Supported Script Types	All server-side scripts
Governance	None
Module	N/url Module
Since	Version 2015 Release 1

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.domain	string	required	The domain name.	Version 2015 Release 1
options.parameters	Object	required	Additional URL parameters as name/value pairs.	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/url Module Script Sample](#).

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)

Method Description	Returns a domain name for a NetSuite account.
Returns	string
Supported Script Types	Client and server-side scripts
Governance	None
Module	N/url Module
Since	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 <code>url.HostType</code> 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.	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 Sample](#).

```
//Add additional code
...
var output = url.resolveDomain({
    hostType: url.HostType.APPLICATION,
    accountId: '012345'
});
...
//Add additional code
```

url.resolveRecord(options)

Method Description	Returns the URL string to a NetSuite record.
Returns	URL to a NetSuite record as a string
Supported Script Types	All server-side scripts
Governance	None
Module	N/url Module
Since	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 <code>true</code> , returns a URL for the record in Edit mode. If set to <code>false</code> , returns a URL for the record in View mode The default value is <code>View</code> .	2015.1
options.params	Object	required	Object used to add parameters for a custom URL. For example, a query to a	2015.1

Parameter	Type	Required / Optional	Description	Since
			database or to a search engine	

Syntax



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 Sample](#).

```
//Add additional code
...
var output = url.resolveRecord({
    recordType: 'salesorder',
    recordId: 6,
    isEditMode: true
    params: p
});
...
//Add additional code
```

url.resolveScript(options)

Method Description	Returns an external or internal URL string to a script.
Returns	The URL as a string
Supported Script Types	All server-side scripts
Governance	None
Module	N/url Module
Since	2015.1

Parameters



Note: The options parameter is a JavaScript object.

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.returnExternalUrl	boolean true false	required	Indicates whether to return the External URL.	2015.1

Parameter	Type	Required / Optional	Description	Since
			By default, the internal URL is returned.	
options.params	Object	required	The object containing name/value pairs to describe the query.	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 Sample](#).

```
//Add additional code
...
var output = url.resolveScript({
   scriptId: 'custom_script',
    deploymentId: 'custom_script_deployment',
    returnExternalUrl: true
    params: p
});
...
//Add additional code
```

url.resolveTaskLink(options)

Method Description	Returns the internal URL to a NetSuite tasklink.
Returns	The URL as a string
Supported Script Types	All server-side scripts
Governance	None
Module	N/url Module
Since	2015.2

Parameters



Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.id	string	required	Internal ID for the tasklink.	2015.1

Parameter	Type	Required / Optional	Description	Since
			<p>i Note: Each page in NetSuite has a unique Tasklink Id associated with it for a specific record type. You can determine the Tasklink for a page within NetSuite by viewing the HTML page source. Search for a string similar to the following, where LIST_SCRIPT refers to the TASKLINK: onclick="nlPopupHelp('LIST_SCRIPT');".</p>	
options.params	Map	optional	The Map object containing name/value pairs to describe the query.	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 Sample](#).

```
//Add additional code
...
u = url.resolveTaskLink('SRCH_JOB', p);
...
//Add additional code
```

url.HostType

Enum Description	Enumeration whose string values each describe a category of domain name. This enum is used to set the value of the hostType parameter of the url.resolveDomain(options) method.
	<p>i Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</p>
Type	enum
Module	N/url Module
Since	2017.1

Values

Value	Description	Sample Result
APPLICATION	The domain for UI access, for users with a non-Customer Center role.	system.na2.netsuite.com
CUSTOMER_CENTER	The domain for UI access, for users with a Customer Center role.	system.na2.netsuite.com
FORM	The domain for forms hosted online, usually in Suitelets.	forms.na2.netsuite.com
RESTLET	The domain for calling a RESTlet from an external source.	rest.na2.netsuite.com
SUITETALK	The domain for SuiteTalk (web services) requests.	webservices.na2.netsuite.com



Warning: The results returned, as shown in the sample results column, may change without notice. Because these values can change, your scripts must dynamically discover domain names. For more details, see the help topic [Understanding Multiple Data Centers](#).

Syntax



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

```
//Add additional code
...
var output = url.resolveDomain({
    hostType: url.HostType.APPLICATION,
    accountId: '012345'
});
...
//Add additional code
```

N/util Module

This module exposes the util Object and its members, made up primarily of methods that verify type on objects and primitives in a SuiteScript 2.0 script.

Each type verification method (for example, `util.isArray(obj)`) returns a boolean value, based on evaluation of the `obj` parameter.

If you need to identify a type specific to SuiteScript 2.0, use the `toString()` global method.



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 and Methods](#).

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

N/util Module Members

Member Type	Name	Return Type / Value Type	Description
Method	util.isArray(obj)	boolean true false	Returns true if the <code>obj</code> parameter is a JavaScript array and false otherwise.
	util.isBoolean(obj)	boolean true false	Returns true if the <code>obj</code> parameter is a Boolean and false otherwise.
	util.isDate(obj)	boolean true false	Returns true if the <code>obj</code> parameter is a JavaScript Date object and false otherwise.
	util.each(iterable, callback)	Object or Array	Iterates over each member in an Object or Array.
	util.extend(receiver, contributor)	Object	Copies the properties in a source object to a destination object and returns the destination object.
	util.isFunction(obj)	boolean true false	Returns true if the <code>obj</code> parameter is a JavaScript Function object and false otherwise.
	util.isNumber(obj)	boolean true false	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.
	utilisObject(obj)	boolean true false	Returns true if the <code>obj</code> parameter is a strictly a JavaScript Object, and false otherwise.
	util.isRegExp(obj)	boolean true false	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.
	util.isString(obj)	boolean true false	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.
	util.nanoTime()	number	Returns the amount of time elapsed from an arbitrary fixed point, in nanoseconds.

N/util Module Script Sample

```
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');
});
// log.debug(rec) //Shows the JSON representation of the current values in a record object
var id = rec.save();
});

```

util.isArray(obj)

Method Description	Returns true if the <code>obj</code> parameter is a JavaScript Array object and false otherwise.
Returns	boolean true false
Supported Script Types	All script types
Governance	None
Module	N/util Module
Global object	util Object
Since	Version 2016 Release 1

Parameters

Parameter	Type	Required / Optional	Description	Since
obj	Object Primitive	Required	Object for which you want to verify the type.	Version 2016 Release 1

Syntax



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

```
//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)

Method Description	Returns true if the <code>obj</code> parameter is a boolean and false otherwise.
Returns	boolean true false
Supported Script Types	All script types
Governance	None
Module	N/util Module
Since	Version 2016 Release 1

Parameters

Parameter	Type	Required / Optional	Description	Since
<code>obj</code>	Object Primitive	Required	Object for which you want to verify the type.	Version 2016 Release 1

Syntax



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

```
//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)

Method Description	Returns true if the <code>obj</code> parameter is a JavaScript Date object and false otherwise.
Returns	boolean true false
Supported Script Types	All script types
Governance	None
Module	N/util Module
Since	Version 2016 Release 1

Parameters

Parameter	Type	Required / Optional	Description	Since
<code>obj</code>	Object Primitive	Required	Object for which you want to verify the type.	Version 2016 Release 1

Syntax



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

```
//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)

Method Description	Returns true if the <code>obj</code> parameter is a JavaScript Function object and false otherwise.
Returns	boolean true false
Supported Script Types	All script types
Governance	None
Module	N/util Module
Since	Version 2016 Release 1

Parameters

Parameter	Type	Required / Optional	Description	Since
obj	Object Primitive	Required	Object for which you want to verify the type.	Version 2016 Release 1

Syntax

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

```
//Add additional code
...
function test() {};
var test2 = function() {};

util.isFunction(test);    // returns true
util.isFunction(test2);   // returns true
...
//Add additional code
```

util.isNumber(obj)

Method Description	Returns true if the obj parameter is a JavaScript Number object or primitive, and false otherwise.
Returns	boolean true false
Supported Script Types	All script types
Governance	None
Module	N/util Module
Since	Version 2016 Release 1

Parameters

Parameter	Type	Required / Optional	Description	Since
obj	Object Primitive	Required	Object for which you want to verify the type.	Version 2016 Release 1

Syntax

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

```
//Add additional code
```

```

...
util.isNumber(112);           // returns true
util.isNumber("112");         // returns false
util.isNumber(NaN);          // returns true

var testNum = 112;
util.isNumber(testNum.valueOf()); // returns true
...
//Add additional code

```

util.isObject(obj)

Method Description	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.
Returns	boolean true false
Supported Script Types	All script types
Governance	None
Module	N/util Module
Since	Version 2016 Release 1

Parameters

Parameter	Type	Required / Optional	Description	Since
<code>obj</code>	Object Primitive	Required	Object for which you want to verify the type.	Version 2016 Release 1

Syntax



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

```

//Add additional code
...
util.isObject({});           // returns true
util.isObject(function() {}); // returns false
...
//Add additional code

```

util.isRegExp(obj)

Method Description	Returns true if the <code>obj</code> parameter is a JavaScript RegExp object, and false otherwise.
---------------------------	--

Returns	boolean true false
Supported Script Types	All script types
Governance	None
Module	N/util Module
Since	Version 2016 Release 1

Parameters

Parameter	Type	Required / Optional	Description	Since
obj	Object Primitive	Required	Object for which you want to verify the type.	Version 2016 Release 1

Syntax



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

```
//Add additional code
...
util.isRegExp(/this is a regexp/); // returns true
util.isRegExp(new RegExp('this is another regexp')); // returns true
...
//Add additional code
```

util.isString(obj)

Method Description	Returns true if the <code>obj</code> parameter is a JavaScript String object or primitive, and false otherwise
Returns	boolean true false
Supported Script Types	All script types
Governance	None
Module	N/util Module
Since	Version 2016 Release 1

Parameters

Parameter	Type	Required / Optional	Description	Since
obj	Object Primitive	Required	Object for which you want to verify the type.	Version 2016 Release 1

Syntax



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

```
//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()

Method Description	Returns the current time (epoch) in nanoseconds. You can use this method to measure elapsed time between two events.
Returns	string
Supported Script Types	Server-side scripts
Governance	None
Module	N/util Module
Since	Version 2016 Release 2

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 call to util.nanoTime()

```
//Add additional code
var startTime = util.nanoTime();
...
var elapsedTime = util.nanoTime() - startTime;
...
//Add additional code
```

util.each(iterable, callback)

Method Description	Iterates over each member in an Object or Array. This method calls the <code>callback</code> function on each member of the iterable.
---------------------------	---

Returns	The original collection as an Object Array
Supported Script Types	All script types
Governance	None
Module	N/util Module
Since	Version 2016 Release 1

Parameters

Parameter	Type	Required / Optional	Description	Since
iterable	Object Array	Required	The data collection to iterate on	Version 2016 Release 1
callback	Function	Required	Takes the custom logic that you want to execute on each member of your collection of data.	Version 2016 Release 1

Syntax

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

```
//Add additional code
...
// Iterate through the object and set the key-value pairs on the record
util.each(itemList, function(quantity, itemId){
    rec.selectNewLine('item');
    rec.setCurrentSublistValue('item','item',itemId);
    rec.setCurrentSublistValue('item','quantity',quantity);
    rec.commitLine('item');
});
...
//Add additional code
```

util.extend(receiver, contributor)

Method Description	Method used to copy the properties in a source object to a destination object. Returns the destination object. You can use this method to merge two objects.
Returns	The Object receiving the properties copied from the contributor
Supported Script Types	All script types
Governance	None
Module	N/util Module
Since	Version 2016 Release 1

Syntax



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

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

This module loads the workflow module to initiate new workflow instances or trigger existing workflow instances.

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

N/workflow Module Members

Member Type	Name	Return Type / Value Type	Description
Method	workflow.initiate(options)	number	Initiates a workflow on-demand. This method is the programmatic equivalent of the Initiate Workflow Action action in SuiteFlow. Returns the internal ID (number) of the workflow instance used to track the workflow against the record.
	workflow.trigger(options)	number	Triggers a workflow on a record. The actions and transitions of the workflow are evaluated for the record in the workflow instance, based on the current state for the workflow instance. Returns the internal ID (number) of the workflow instance used to track the workflow against the record.

N/workflow Module Script Sample

The following example searches for a specific workflow deployed on the customer record and then executes it.

This sample script uses the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see [SuiteScript 2.0 – Script Architecture](#) and [SuiteScript 2.0 Script Types and Entry Points](#).



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)

Method Description	Initiates a workflow on-demand. This method is the programmatic equivalent of the Initiate Workflow Action action in SuiteFlow. Returns the internal ID of the workflow instance used to track the workflow against the record.
Returns	number
Supported Script Types	All server-side scripts
Governance	20 usage units
Module	N/workflow Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.recordType	number	required	The record type ID of the workflow base record. For example, use 'customer', 'salesorder', or 'lead'. This is the Record Type field on the Workflow Definition Page .	Version 2015 Release 2
options.recordId	string number	required	The internal ID of the base record	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.workflowId	string number	required	The internal ID (number) or script ID (string) for the workflow definition. This is the ID field on the Workflow Definition Page .	Version 2015 Release 2
options.defaultValue	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 Workflow Definition Page or workflow and state Workflow Custom Fields .	Version 2015 Release 2

Syntax



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)

Method Description	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.
Returns	number
Supported Script Types	All server-side scripts
Governance	20 usage units
Module	N/workflow Module
Since	Version 2015 Release 2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.recordType	number	required	The record type ID of the workflow base record. For example, use 'customer', 'salesorder', or 'lead'. This is the Record Type field on the Workflow Definition Page .	Version 2015 Release 2
options.recordId	string number	required	The internal ID of the base record	Version 2015 Release 2
options.workflowId	string number	required	The internal ID (number) or script ID (string) for the workflow definition. This is the ID field on the Workflow Definition Page .	Version 2015 Release 2
options.workflowInstanceId	string number	optional	The internal ID of the workflow instance.	Version 2015 Release 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.	Version 2015 Release 2
options.stateId	string number	optional	The internal ID (number) or script ID (string) of the workflow instance.	Version 2015 Release 2

Syntax

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

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 Sample](#)

N/xml Module Members

Member Type	Name	Return Type / Value Type	Description
Object	xml.Parser	Object	Encapsulates the functionality used by NetSuite to parse XML.
	xml.XPath	Object	Encapsulates the functionality used by NetSuite to run XPath expressions. XPath is a standard for enumerating paths in an XML document collection.
	xml.Node	Object	Represents a generic XML node in an XML document. A node can be a Document, Element, or Attribute.
	xml.Document	Object	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.
	xml.Element	Object	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.

Member Type	Name	Return Type / Value Type	Description
	xml.Attr	Object	Represents an attribute node of an xml.Element object.
Method	xml.escape(options)	string	Prepares a string for use in XML by escaping XML markup, such as angle brackets, quotation marks, and ampersands.
	xml.validate(options)	void	Validates an XML document against an XML Schema (XSD).
Enum	xml.NodeType	string (read-only)	Enumeration that holds the string values for the supported node types. The Node.nodeType property is defined by one of the values in this enum.

Parser Object Members

The following members are called on the [xml.Parser](#) object.

Member Type	Name	Return Type / Value Type	Description
Method	Parser.fromString(options)	xml.Document	Parses a string into a W3C XML document object.
	Parser.toString(options)	string	Converts (serializes) an xml.Document object into a string.

XPath Object Members

The following members are called on the [xml.XPath](#) object.

Member Type	Name	Return Type / Value Type	Description
Method	XPath.select(options)	xml.Node[]	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	Description
Method	Node.appendChild(options)	xml.Node	Appends a node after the last child node of a specific element node. Returns the new child node.
	Node.cloneNode(options)	xml.Node	Creates a copy of a node. Returns the copied node.

Member Type	Name	Return Type / Value Type	Description
	Node.compareDocumentPosition(options)	number	Returns a number that reflects where two nodes are located, compared to each other.
	Node.hasAttributes()	boolean true false	Returns <code>true</code> if the current node has child nodes or returns <code>false</code> if the current node does not have child nodes.
	Node.hasChildNodes()	boolean true false	Returns <code>true</code> if the current node has any attributes. Note that only element nodes can have attributes.
	Node.insertBefore(options)	xml.Node	Inserts a new child node before an existing child node for the current node.
	Node.isDefaultNamespace(options)	boolean true false	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.
	Node.isEqualNode(options)	boolean true false	Returns <code>true</code> if two nodes are equal or returns <code>false</code> if two nodes are not equal.
	Node.isSameNode(options)	boolean true false	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.
	Node.lookupNamespaceURI(options)	string	Returns the namespace uniform resource identifier (URI) that matches the specified namespace prefix.
	Node.lookupPrefix(options)	string	Returns the namespace prefix associated with the specified namespace uniform resource identifier (URI).
	Node.normalize()	void	Puts all text nodes underneath a node, including attribute nodes, into a normal form.
	Node.removeChild(options)	xml.Node	Removes the specified child node. Returns the removed child node.
	Node.replaceChild(options)	xml.Node	Replaces a specific child node with another child node in a list of child nodes.
Property	Node.attributes	Object (read-only)	Key-value pairs for all attributes for an <code>xml.Element</code>

Member Type	Name	Return Type / Value Type	Description
			node. Returns <code>null</code> for all other node types.
	Node.baseURI	string (read-only)	Absolute base uniform resource identifier (URI) of a node or <code>null</code> if the URI cannot be determined.
	Node.childNodes	xml.Node[] (read-only)	Array of all child nodes of a node or an empty array if there are no child nodes.
	Node.firstChild	xml.Node (read-only)	First child node for a specific node or <code>null</code> if there are no child nodes.
	Node.lastChild	xml.Node (read-only)	Last child node for a specific node or <code>null</code> if there is no last child node.
	Node.localName	string (read-only)	The local part of the qualified name of a node.
	Node.namespaceURI	string (read-only)	The namespace uniform resource identifier (URI) of a node or <code>null</code> if there is no namespace URI for the node.
	Node.nextSibling	xml.Node (read-only)	The next node in a node list or <code>null</code> if the current node is the last node.
	Node.nodeName	string (read-only)	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.
	Node.nodeType	string	The type of node defined as a value from the <code>xml.NodeType</code> enum.
	Node.nodeValue	string	The value of a node, depending on its type.
	Node.ownerDocument	xml.Document (read-only)	The root element for a node as a <code>xml.Document</code> object.
	Node.parentNode	xml.Node (read-only)	The parent node of a node.
	Node.prefix	string	The namespace prefix of the node, or <code>null</code> if the node does not have a namespace.
	Node.previousSibling	xml.Node (read-only)	The previous node in a node list or <code>null</code> if the current node is the first node.
	Node.textContent	string	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	Description
Method	<code>Document.adoptNode(options)</code>	<code>xml.Node</code>	Attempts to adopt a node from another document to this document.
	<code>Document.createAttribute(options)</code>	<code>xml.Attr</code>	Creates an attribute node of type ATTRIBUTE_NODE with the optional specified value.
	<code>Document.createAttributeNS(options)</code>	<code>xml.Attr</code>	Creates an attribute node of type ATTRIBUTE_NODE, with the specified namespace value and optional specified value.
	<code>Document.createCDATASection(options)</code>	<code>xml.Node</code>	Creates a CDATA section node of type DOCUMENT_FRAGMENT_NODE with the specified data.
	<code>Document.createComment(options)</code>	<code>xml.Node</code>	Creates a Comment node of type COMMENT_NODE with the specified string.
	<code>Document.createDocumentFragment()</code>	<code>xml.Node</code>	Creates a node of type DOCUMENT_FRAGMENT_NODE.
	<code>Document.createElement(options)</code>	<code>xml.Element</code>	Creates a new node of type ELEMENT_NODE with the specified name.
	<code>Document.createElementNS(options)</code>	<code>xml.Element</code>	Creates a new node of type ELEMENT_NODE with the specified namespace URI and name.
	<code>Document.createProcessingInstruction(options)</code>	<code>xml.Node</code>	Creates a new node of type PROCESSING_INSTRUCTION_NODE with the specified target and data.
	<code>Document.createTextNodeNode(options)</code>	<code>xml.Node</code>	Creates a new node of type TEXT_NODE.
	<code>Document.getElementByNodeId(options)</code>	<code>xml.Element</code>	Returns the element that has an ID attribute with the specified value as an <code>xml.Element</code> object.
	<code>Document.getElementsByTagName(options)</code>	<code>xml.Element[]</code>	Returns an array of <code>xml.Element</code> objects with a specific tag name, in the order in which they appear in the XML document.
	<code>Document.getElementsByTagNameNS(options)</code>	<code>xml.Element[]</code>	Returns an array of <code>xml.Element</code> objects with a specific tag name

Member Type	Name	Return Type / Value Type	Description
			and namespace, in the order in which they appear in the XML document.
	Document.importNode(options)	xml.Node	Imports a node from another document to this document. Creates a new copy of the source node.
Property	Document.doctype	Object (read-only)	Returns a node of type DOCUMENT_TYPE_NODE that represents the doctype of the XML document.
	Document.documentElement	xml.Element (read-only)	Root node of the XML document.
	Document.documentElementURI	string (read-only)	Location of the document or null if undefined.
	Document.inputEncoding	string (read-only)	Encoding used for an XML document at the time the document was parsed.
	Document.xmlEncoding	string (read-only)	Part of the XML declaration, the XML encoding of the XML document.
	Document.xmlStandalone	boolean true false	Part of the XML declaration, returns true if the current XML document is standalone or returns false if it is not.
	Document.xmlVersion	string	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	Description
Method	Element.getAttribute(options)	string	Returns the value of the specified attribute.
	Element.getAttributeNode(options)	xml.Attr	Retrieves an attribute node by name.
	Element.getAttributeNodeNS(options)	string	Returns an attribute node with the specified namespace URI and local name.

Member Type	Name	Return Type / Value Type	Description
	Element.getAttributeNS(options)	xml.Attr	Returns an attribute value with the specified namespace URI and local name.
	Element.getElementsByTagName(options)	xml.Element[]	Returns an array of descendant <code>xml.Element</code> objects with a specific tag name, in the order in which they appear in the XML document.
	Element.getElementsByTagNameNS(options)	xml.Element[]	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 true false	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 true false	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	Removes the attribute with the specified name.
	Element.removeAttributeNode(options)	xml.Attr	Removes the attribute specified as a <code>xml.Attr</code> object.
	Element.removeAttributeNS(options)	void	Removes the attribute with the specified namespace URI and local name.
	Element.setAttribute(options)	void	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	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	Adds the specified attribute node. If an attribute with the same local name and namespace URI is already

Member Type	Name	Return Type / Value Type	Description
			present in the element, it is replaced by the new one.
	Element.setAttributeNS(options)	void	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)	The tag name of this xml.Element object.

Attr Object Members

The following members are called on the [xml.Attr](#) object.

Member Type	Name	Return Type / Value Type	Description
Property	Attr.name	string (read-only)	The name of an attribute.
	Attr.ownerElement	xml.Element (read-only)	The xml.Element object that is the parent of the xml.Attr object.
	Attr.specified	boolean true false	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.
	Attr.value	string	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 Sample

The N/xml module sample references the following XML file, 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>

```

The following Suitelet example loads an XML file from the file cabinet, iterates through the individual book nodes, and accesses the child node values using two common methods: (through firstChild/nextSibling/etc and through getElementsByTagName)

```

/**
 * @NApiVersion 2.x
 * @NScriptType Suitelet
 */
define(['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 : '//b:book'
                });

                for (var i = 0; i < bookNode.length; i++) {
                    var title = bookNode[i].firstChild.nextSibling.textContent;
                    var author = bookNode[i].getElementsByTagName({
                        tagName : 'b:author'
                    })[0].textContent;
                    sentence += 'Author: ' + author + ' wrote ' + title + '\n';
                }
                options.response.write(sentence);
            }
        };
});

```

The following output is produced from the sample code when used with the BookSample.xml document:

Author: Giada De Laurentiis wrote *Everyday Italian*.
 Author: J K. Rowling wrote *Harry Potter*.
 Author: James McGovern wrote *XQuery Kick Start*.
 Author: Erik T. Ray wrote *Learning XML*.

xml.Parser

Object Description	Encapsulates the functionality used by NetSuite to parse an XML document. For a complete list of this object's methods, see Parser Object Members .
Supported Script Types	All script types
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var parserObj = xml.Parser;
...
//Add additional code
```

Parser.fromString(options)

Method Description	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.
	Note: 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.
Returns	<code>xml.Document</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.text	string	Required	String being converted to an xml.Document .

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)

Method Description	Converts (serializes) an xml.Document object into a string. This API is useful, for example, if you want to serialize and store an xml.Document in a custom field.
Returns	string
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.document	xml.Document	Required	XML document being serialized.

Syntax

```
//Add additional code
```

```

...
var xmlStringContent = xml.Parser.toString({
    document : xmlDocument
});
...
//Add additional code

```

xml.XPath

Object Description	Encapsulates the functionality to run XPath expressions. For a complete list of this object's methods, see XPath Object Members .
Supported Script Types	All script types
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

//Add additional code
...
var xpath = xml.XPath;
...
//Add additional code

```

XPath.select(options)

Method Description	Selects an array of nodes from an XML that match an XPath expression.
Returns	<code>xml.Node[]</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.node	<code>xml.Node</code>	Required	XML node being queried.
options.xpath	<code>string</code>	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

Object Description	Represents a single node in an XML document tree. The XML DOM presents a document as a hierarchy of node objects. See the xml.NodeType enum for a list of possible node types. NetSuite supports a subset of W3C DOM methods. For a complete list of this object's methods and properties, see Node Object Members .
Supported Script Types	All script types
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var bookNode = xml.XPath.select({
    node : xmlDocument,
    xpath : '//book'
});
...
//Add additional code
```

Node.appendChild(options)

Method Description	Appends a node after the last child node of a specific element node. Returns the new child node.
Returns	xml.Node
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

i Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.newChild	xml.Node	Required	xml.Node 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 newnode = elem[0].appendChild({
    newChild : node
});
...
//Add additional code
```

Node.cloneNode(options)

Method Description	Creates a copy of a node. Returns the copied node.
Returns	xml.Node
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

i Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.deep	boolean true false	Optional	Use true to clone the node, attributes, and all descendants. Use false to only clone the node and attributes.

Syntax

```
//Add additional code
```

```

...
var copiednode = elem[0].cloneNode({
    deep : true
});
...
//Add additional code

```

Node.compareDocumentPosition(options)

Method Description	Returns a number that reflects where two nodes are located, compared to each other. Returns one of the following numbers: <ul style="list-style-type: none"> ■ 1. The two nodes do not belong to the same document. ■ 2. The specified node comes before the current node. ■ 4. The specified node comes after the current node. ■ 8. The specified node contains the current node. ■ 16. The current node contains the specified node. ■ 32. The specified and current nodes do not have a common container node or the two nodes are different attributes of the same node. <p>Note: The return value can be a combination of the above values. For example, a return value of 20 means the specified node is contained by the current node, a value of 16, and the specified node follows the current node, a value of 4.</p> <p>Important: This method is not supported on Internet Explorer.</p>
Returns	number
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
options.other	xml.Node	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 xml.Node.

Syntax

```
//Add additional code
...
var posCode = elem[0].compareDocumentPosition({
    other : parentNode[0]
});
...
//Add additional code
```

Node.hasAttributes()

Method Description	Returns <code>true</code> if the current node has attributes defined, or <code>false</code> otherwise.
	 Important: This method is not supported on Internet Explorer.
Returns	<code>boolean true false</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var hasAttributes = parentNode[0].hasAttributes()
...
//Add additional code
```

Node.hasChildNodes()

Method Description	Returns <code>true</code> if the current node has child nodes or returns <code>false</code> if the current node does not have child nodes.
Returns	<code>boolean true false</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
```

```

...
var hasChildren = parentNode[0].hasChildNodes()
...
//Add additional code

```

Node.insertBefore(options)

Method Description	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.
Returns	xml.Node
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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 <code>refChild</code> is <code>null</code> , 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)

Method Description	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 Node.namespaceURI .
Returns	<code>boolean true false</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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)

Method Description	Returns <code>true</code> if two nodes are equal or returns <code>false</code> if two nodes are not equal. The two nodes are equal if they meet the following conditions: <ul style="list-style-type: none"> ■ Both nodes have the same type. ■ Both nodes have the same attributes and attribute values. The order of the attributes is not considered. ■ Both nodes have equal lists of child nodes and the child nodes appear in the same order.
---------------------------	---

	<p>Note: Two nodes may be equal, even if they are not the same. See <code>Node.isSameNode(options)</code>.</p>
	<p>Important: This method is not supported on Internet Explorer.</p>
Returns	<code>boolean true false</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.other</code>	<code>xml.Node</code>	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)

Method Description	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>Note: Two nodes that are the same are also equal. See <code>Node.isEqualNode(options)</code>.</p>
	<p>Important: This method is not supported on Internet Explorer or Firefox.</p>
Returns	<code>boolean true false</code>
Supported Script Types	All script types

Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
options.other	xml.Node	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)

Method Description	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.
Important:	This method is not supported on Internet Explorer.
Returns	string
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

 **Note:** 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)

Method Description	Returns the namespace prefix associated with the specified namespace uniform resource identifier (URI). Returns <code>null</code> 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.
	 Important: This method is not supported on Internet Explorer.
Returns	<code>string</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.namespaceURI</code>	<code>string</code>	Required	Namespace URI associated with the namespace prefix.

Syntax

```
//Add additional code
...
var prefix = parentNode[0].lookupPrefix({
    namespaceURI : '*'
});
...
//Add additional code
```

Node.normalize()

Method Description	Puts all text nodes underneath a node, including attribute nodes, into a normal form. In normal form, only structure (such as elements, comments, processing instructions, CDATA sections, and entity references) separates text nodes. After normalization, there are no adjacent or empty text nodes. Use this method if you require a particular document tree structure and want to make sure that the XML DOM view of a document is identical when you save and reload it.
Returns	void
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
node.normalize();
...
//Add additional code
```

Node.removeChild(options)

Method Description	Removes the specified child node.
Returns	xml.Node
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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)

Method Description	Replaces a specific child node with another child node in a list of child nodes. If the new child node to add already exists in the list of child nodes, the node is first removed.
Returns	xml.Node
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
options.newChild	xml.Node	Required	New child node to add.
options.oldChild	xml.Node	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.

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.	Child node cannot be replaced.

Syntax

```
//Add additional code
...
var replacednode = parentNode.replaceChild({
    newChild : elem[2],
    oldChild : elem[1]
});
...
//Add additional code
```

Node.attributes

Property Description	Key-value pairs for all attributes for an <code>xml.Element</code> node. Returns <code>null</code> for all other node types.
Type	string (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var attrbs = elem[0].attributes;
...
//Add additional code
```

Node.baseURI

Property Description	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> .
	Note: The format of this value is browser-specific.
Type	string (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var baseuri = parentNode[0].baseURI;
...
//Add additional code
```

Node.childNodes

Property Description	Array of all child nodes of a node or an empty array if there are no child nodes.
Type	xml.Node[]
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var childnodes = parentNode[0].childNodes;
...
//Add additional code
```

Node.firstChild

Property Description	The first child node of a node, or <code>null</code> if there are no child nodes.
Type	xml.Node
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var nodeValue1 = bookNode[0].firstChild.nextSibling.textContent;
...
//Add additional code
```

Node.lastChild

Property Description	The last child node of a node, or <code>null</code> if there are no child nodes.
-----------------------------	--

Type	xml.Node
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var nodeValue = parentNode[0].lastChild.previousSibling.textContent;
...
//Add additional code
```

Node.localName

Property Description	The local part of the qualified name of a node.
Type	string (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var localname = parentNode[0].localName;
...
//Add additional code
```

Node.namespaceURI

Property Description	The namespace uniform resource identifier (URI) of a node or <code>null</code> if there is no namespace URI for the node.
Type	string (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var uri = parentNode[0].namespaceURI;
...
```

```
//Add additional code
```

Node.nextSibling

Property Description	The next node in a node list or <code>null</code> if the current node is the last node.
Type	<code>xml.Node</code> (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var nodeName = parentNode[0].firstChild.nextSibling.textContent;
...
//Add additional code
```

Node.nodeName

Property Description	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.
	Note: On Chrome, this property also includes the namespace or prefix.
Type	<code>string</code> (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var nodeName = parentNode[0].firstChild.nodeName;
...
//Add additional code
```

Node.nodeType

Property Description	The type of node as an enum. For all possible values of this property, see <code>xml.NodeType</code> .
-----------------------------	---

Type	xml.NodeType
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var nodeType = parentNode[0].firstChild.nodeType;
...
//Add additional code
```

Node.nodeValue

Property Description	The value of a node, depending on its type. If the value is <code>null</code> , setting this value has no effect.
Type	string
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var nodeValue = parentNode[0].firstChild.nodeValue;
...
//Add additional code
```

Node.ownerDocument

Property Description	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> .
Type	<code>xml.Document</code>
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

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

```

var doc = parentNode[0].ownerDocument;
...
//Add additional code

```

Node.parentNode

Property Description	The parent node of a node. All node types, except <code>xml.Attr</code> , <code>xml.Document</code> , <code>DocumentFragment</code> , <code>Entity</code> , and <code>Notation</code> can have a parent node. See <code>xml.NodeType</code> for possible node types.
Type	<code>xml.Node</code>
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

//Add additional code ...
var nodevalue = parentNode[0].lastChild.parentNode.textContent;
...
//Add additional code

```

Node.prefix

Property Description	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.
Type	<code>string</code>
Module	N/xml Module
Since	Version 2015 Release 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

Property Description	The previous node in a node list or <code>null</code> if the current node is the first node.
Type	<code>xml.Node</code>
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var nodeValue = parentNode[0].lastChild.previousSibling.textContent;
...
//Add additional code
```

Node.textContent

Property Description	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.
Type	<code>string</code>
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var nodeValue = parentNode[0].firstChild.textContent;
...
//Add additional code
```

xml.Document

Object Description	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 <code>xml.Document</code> object to manipulate the XML document and the nodes in the document tree.
---------------------------	--

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

An XML document object is also a node of type DOCUMENT_NODE. In addition to the Document object members, Document objects inherit the members of the Node object. For a complete list of these methods and properties, see [Node Object Members](#).

Supported Script Types	All script types
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var xmlDocument = xml.Parser.fromString({
    text : xmlFileContent
});
...
//Add additional code
```

Document.adoptNode(options)

Method Description	Attempts to adopt a node from another document to this document. If successful, this method changes the Node.ownerDocument property of the source node, its children, and any attribute nodes to the current document. If the source node has a parent node, the parent node is first removed from the child list of its own parent node.
	 Important: This method is not supported on Internet Explorer.
Returns	xml.Node
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
options.source	xml.Node	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)

Method Description	Creates an attribute node of type ATTRIBUTE_NODE with the optional specified value and returns the new <code>xml Attr</code> object. The localName, prefix, and namespaceURI properties of the new node are set to null.
Returns	<code>xml Attr</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

Note: 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)

Method Description	Creates an attribute node of type ATTRIBUTE_NODE, with the specified namespace value and optional specified value, and returns the new <code>xml.Attr</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> .
Returns	<code>xml.Attr</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.namespaceURI</code>	string	Required	Namespace URI of the attribute to create. Value can be <code>null</code> .
<code>options.qualifiedName</code>	string	Required	Qualified name of the new attribute node.
<code>options.value</code>	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)

Method Description	Creates a CDATA section node of type DOCUMENT_FRAGMENT_NODE with the specified data and returns the new xml.Node object.
Returns	xml.Node
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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)

Method Description	Creates a Comment node of type COMMENT_NODE with the specified string.
Returns	xml.Node
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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()

Method Description	Creates a node of type DOCUMENT_FRAGMENT_NODE and returns the new xml.Node object.
Returns	xml.Node
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var newNode = xmlDocument.createDocumentFragment();
```

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

Document.createElement(options)

Method Description	Creates a new node of type ELEMENT_NODE with the specified name and returns the new <code>xml.Element</code> 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> .
Returns	<code>xml.Element</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

i 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 <code>tagName</code> value.

Syntax

```
//Add additional code
...
var elem = xmlDocument.createElement({
    tagName : 'book'
});
...
//Add additional code
```

Document.createElementNS(options)

Method Description	Creates a new node of type ELEMENT_NODE with the specified namespace URI and name and returns the new <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> .
Returns	<code>xml.Element</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description
<code>options.namespaceURI</code>	string	Required	Namespace URI of the element to create. Can be <code>null</code> .
<code>options.qualifiedName</code>	string	Required	Qualified 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 with the specified namespace cannot be created.

Syntax

```
//Add additional code
...
var elem = xmlDocument.createElementNS({
    namespaceURI : '*',
    qualifiedName : 'book'
});
...
//Add additional code
```

Document.createProcessingInstruction(options)

Method Description	Creates a new node of type <code>PROCESSING_INSTRUCTION_NODE</code> with the specified target and data and returns the new <code>xml.Node</code> object. The following example shows a sample processing instruction: <code><?xml version="1.0"?></code> Use a processing instruction node to keep processor-specific information in the text of the XML document.
---------------------------	--

Returns	xml.Node
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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)

Method Description	Creates a new text node and returns the new xml.Node object.
Returns	xml.Node
Supported Script Types	All script types
Governance	None
Module	N/xml Module

Since	Version 2015 Release 2
-------	------------------------

Parameters

i 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)

Method Description	Returns the element that has an ID attribute with the specified value as an <code>xml.Element</code> object. Returns <code>null</code> if no such element exists.
Returns	<code>xml.Element</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

i 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)

Method Description	Returns an array of <code>xml.Element</code> objects with a specific tag name, in the order in which they appear in the XML document.
Returns	<code>xml.Element[]</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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)

Method Description	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.
Important: This method is not supported on Internet Explorer.	
Returns	<code>xml.Element[]</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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)

Method Description	Imports a node from another document to this document. This method creates a new copy of the source node. If the <code>deep</code> parameter is set to <code>true</code> , it imports all children of the specified node. If set to <code>false</code> , it imports only the node itself. Method returns the imported xml.Node object.
Returns	xml.Node
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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.

Parameter	Type	Required / Optional	Description
options.deep	boolean true false	Required	<p>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.</p> <div style="border: 1px solid #f0e68c; padding: 5px; margin-top: 10px;">  Important: This parameter is not supported on Internet Explorer. </div>

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

Property Description	The doctype of the XML document.
	<div style="border: 1px solid #f0e68c; padding: 5px;">  Important: This property is not supported on Internet Explorer. </div>
Type	xml.Element (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var doctype = xmlDocument.doctype;
...
//Add additional code
```

Document.documentElement

Property Description	Root node of the XML document. Use this property to directly access the <code>xml.Element</code> object that represents the root node of an XML document.
Type	<code>xml.Element</code> (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var root = xmlDocument.documentElement;
...
//Add additional code
```

Document.documentElementURI

Property Description	Location of the document or <code>null</code> if undefined.
Type	<code>string</code>
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var documentURI = xmlDocument.documentElementURI;
...
//Add additional code
```

Document.inputEncoding

Property Description	Encoding used for an XML document at the time the document was parsed. When parsing an XML document with the following declaration, the <code>inputEncoding</code> property is UTF-8: <code><?xml version="1.0" encoding="UTF-8" standalone="yes"?></code>
Type	<code>string</code> (read-only)

Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var encoding = xmlDocDocument.inputEncoding;
...
//Add additional code
```

Document.xmlEncoding

Property Description	Part of the XML declaration, the XML encoding of the XML document. In the following declaration, the xmlEncoding property is UTF-8: <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
Important:	This property is not supported on Internet Explorer or Firefox.
Type	string (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var encoding = xmlDocDocument.xmlEncoding;
...
//Add additional code
```

Document.xmlStandalone

Property Description	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. In the following declaration, the xmlStandalone property is <code>true</code> : <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
Important:	This property is not supported on Internet Explorer or Firefox.
Type	boolean <code>true</code> <code>false</code>
Module	N/xml Module

Since

Version 2015 Release 2

Syntax

```
//Add additional code
...
var isStandalone = xmlDocument.xmlStandalone;
...
//Add additional code
```

Document.xmlVersion

Property Description	Part of the XML declaration, the version number of the XML document. In the following declaration, the xmlVersion property is 1.0: <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
Type	string (read-only)
Module	N/xml Module
Since	Version 2015 Release 2



Important: This property is not supported on Internet Explorer or Firefox.

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

Object Description	Represents an element in an XML document. Elements may contain attributes, other elements, or text. If an element contains text, the text is represented in a text node of type TEXT_NODE.
---------------------------	--

For example, the following element year contains a text node with the value of 2015:

```
<year>2015</year>
```

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

An XML element object is also a node of type ELEMENT_NODE. In addition to the Element object members, Element objects inherit the members of the Node object. For a complete list of these methods and properties, see [Node Object Members](#).

Supported Script Types	All script types
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var elem = parentNode[0].getElementsByTagName({tagName : 'title'});
...
//Add additional code
```

Element.getAttribute(options)

Method Description	Returns the value of the specified attribute.
Returns	xml.Attr
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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)

Method Description	Retrieves an attribute node by name.
	 Important: This method is not supported on Internet Explorer.
Returns	xml.Attr
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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)

Method Description	Returns an attribute node with the specified namespace URI and local name.
	 Important: This method is not supported on Internet Explorer.
Returns	string

Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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 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)

Method Description	Returns an attribute value with the specified namespace URI and local name.  Important: This method is not supported on Internet Explorer.
Returns	string

Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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)

Method Description	Returns an array of descendant <code>xml.Element</code> objects with a specific tag name, in the order in which they appear in the XML document.
Returns	<code>xml.Element[]</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module

Since

Version 2015 Release 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)

Method Description	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.
	 Important: This method is not supported on Internet Explorer.
Returns	<code>xml.Element[]</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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.

Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	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)

Method Description	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> .
	⚠ Important: This method is not supported on Internet Explorer.
Returns	<code>boolean true false</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

i 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)

Method Description	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> .
	Important: This method is not supported on Internet Explorer.
Returns	<code>boolean true false</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

i Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
<code>options.namespaceURI</code>	string	Required	Namespace URI of the attribute to match on.
<code>options.localName</code>	string	Required	Local name of the attribute to match on.

Errors

Error Code	Message	Thrown If
<code>SSS_XML_DOM_EXCEPTION</code>	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)

Method Description	Removes the attribute with the specified name.
Returns	void
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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)

Method Description	Removes the attribute specified as a xml Attr object.
Returns	xml Attr
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.oldAttr	xml.Attr	Required	xml.Attr 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)

Method Description	Removes the attribute with the specified namespace URI and local name. Important: This method is not supported on Internet Explorer.
Returns	void
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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.

Parameter	Type	Required / Optional	Description
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)

Method Description	Adds a new attribute with the specified name. If an attribute with that name is already present in the element, its value is changed to the value specified in method argument. If an attribute with the specified name already exists, the value of the attribute is changed to the value of the value parameter.
Returns	void
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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)

Method Description	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.
Returns	<code>xml Attr</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

i 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
SSS_XML_DOM_EXCEPTION	INUSE_ATTRIBUTE_ERR: 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)

Method Description	Adds the specified attribute node. If an attribute with the same local name and namespace URI is already present in the element, it is replaced by the new one. If an attribute with the same namespaceURI and localName property already exist, it is replaced with the object in the newAttr parameter. If the attribute node replaces an existing attribute node, the method returns the new <code>xml Attr</code> object.
Returns	<code>xml Attr</code>
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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)

Method Description	Adds a new attribute with the specified name and namespace URI. If an attribute with the same name and namespace URI is already present in the element, its value is changed to the value specified in method argument. If an attribute with the specified name already exists, the value of the attribute is changed to the value of the value parameter. If the attribute node replaces an existing attribute node, the method returns the new xml Attr object.
Returns	void
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 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

Property Description	The tag name of this xml.Element object.
Type	string (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

//Add additional code
...
var tagName = elem[0].tagName; \\ returns 'title'.
...
//Add additional code

```

xml.Attr

Object Description	Represents an attribute node of an xml.Element object. For a complete list of this object's properties, see Attr Object Members .
Supported Script Types	All script types
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

//Add additional code
...
var attr = elem[0].getAttributeNode({
    name : 'lang'
});
...
//Add additional code

```

Attr.name

Property Description	The name of an attribute. This property is a qualified name if the <code>Node.localName</code> property for the parent <code>xml.Element</code> object is null.
Type	string (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var attrName = attr.name; \\ returns 'lang'.
...
//Add additional code
```

Attr.ownerElement

Property Description	<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.
Type	<code>xml.Element</code> (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
//Add additional code
...
var attrElement = attr.ownerElement; \\ returns the title element.
...
//Add additional code
```

Attr.specified

Property Description	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.
Type	<code>boolean true false</code>
Module	N/xml Module

Since

Version 2015 Release 2

Syntax

```
//Add additional code
...
var attrSpecified = attr.specified;
...
//Add additional code
```

Attr.value

Property Description	Value of an attribute. The value of the attribute is returned as a string. Character and general entity references are replaced with their values. For example, a character reference such as or an entity reference such as is replaced with a non-breaking space.
Type	string
Module	N/xml Module
Since	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	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)

Method Description	Prepares a string for use in XML by escaping XML markup, such as angle brackets, quotation marks, and ampersands.
---------------------------	---

Returns	string
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.xmlText	string	Required	String being escaped.	Version 2015 Release 2

Syntax

```
//Add additional code
...
var xmlEscapedDocument = xml.escape({
    xmlText : xmlFileContent
});
...
//Add additional code
```

xml.validate(options)

Method Description	Validates an XML document against an XML Schema (XSD).
	 Important: This method only validates XML Schema (XSD); validation of other XML schema languages is not supported.
	The XML document must be passed as an <code>xml.Document</code> object. The location of the source XML Document does not matter; the validation is performed with the Document object stored in memory. The XSD must be stored in the File Cabinet.
Returns	void
Supported Script Types	All server-side script types
Governance	None
Module	N/xml Module
Since	Version 2015 Release 2

Parameters

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

Parameter	Type	Required / Optional	Description	Since
options.xml	xml.Document	Required	The xml.Document object to validate.	Version 2015 Release 2
options.xsdFilePathOrId	number string	Required	The file ID or path to the XSD in the File Cabinet to validate the XML document against.	Version 2015 Release 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.	Version 2015 Release 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 : xmlDoc,
    xsdFilePathOrId : 'SuiteScripts/schema_parent.xsd',
    importFolderPathOrId : 'SuiteScripts/'
});
...
//Add additional code
```

xml.NodeType

Enum Description	Enumeration that holds the string values for the supported node types. The Node.nodeType property is defined by one of the values in this enum. Use this enum to determine the type of a node in an XML document.
-------------------------	---

	 Note: Enum values are constants and therefore read-only.
Module	N/xml Module
Since	Version 2015 Release 2

Values

- | | | |
|--|--|--|
| <ul style="list-style-type: none"> ■ ATTRIBUTE_NODE ■ CDATA_SECTION_NODE ■ COMMENT_NODE ■ DOCUMENT_FRAGMENT_NODE | <ul style="list-style-type: none"> ■ DOCUMENT_NODE ■ DOCUMENT_TYPE_NODE ■ ELEMENT_NODE ■ ENTITY_NODE | <ul style="list-style-type: none"> ■ ENTITY_REFERENCE_NODE ■ NOTATION_NODE ■ PROCESSING_INSTRUCTION_NODE ■ TEXT_NODE |
|--|--|--|

Syntax

```
//Add additional code
...
var DocType = xmlDocument.nodeType; \\ returns DOCUMENT_NODE
...
//Add additional code
```

SuiteScript 2.0 JSDoc Validation

JSDoc 3 is a documentation generator for JavaScript source code. Users typically employ this tool to generate an HTML API reference. Developers insert specific comment blocks into their source code. These comment blocks start with `/**` and end with `*/`.

JSDoc also includes its own markup language, which is made up of JSDoc tags. These tags are prefaced with the `@` symbol. The JSDoc tags are added to the comment blocks, and are used to specify the various entities of a JavaScript file (for example, `@param`).

```
/**
 * Creates a file.
 * @param {string} name [required] - file name
 * @param {string} fileType [required] - file type
 * @param {string} contents [required] - file content
 * @returns {object} file.File
 */
```

JSDoc parses the source code for each comment block. The HTML output is then generated based on the content of the comment blocks and an evaluation of the code.

JSDoc 3 also provides users with the ability to create custom JSDoc tags. These tags can be defined to trigger events (for example, displaying a certain page).

SuiteScript 2.0 includes two custom tags, one of which is required in each entry point script uploaded to NetSuite (see [SuiteScript 2.0 JSDoc Tags](#)). When a SuiteScript 2.0 script record is requested, NetSuite uses JSDoc 3 to evaluate the entry point script and parse the code for the required JSDoc tag. This tag is used to validate the SuiteScript version.

 **Note:** SuiteScript 2.0 users can use JSDoc 3 to create their own documentation for scripts, custom modules, and SuiteApps. To take advantage of this tool, developers must download JSDoc 3 from the official website. For additional information on JSDoc 3, see <http://usejsdoc.org/>.

JSDoc Comment Blocks

JSDoc tag comment blocks must start with a `/**` and end with a `*/` to be recognized.

Valid Example:

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

Invalid Examples:

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

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

JSDoc tags consist of a key/value pair. The key ends with the first white space after a string starting with an `@`. The value starts with the next non-whitespace character, and ends with the next carriage return. Each comment line within the block is prefixed with an `*`.

Valid Example:

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

Invalid Example:

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

SuiteScript 2.0 JSDoc Tags

The following table describes the available SuiteScript 2.0 JSDoc tags. Note that SuiteScript 2.0 entry point scripts must include two tags: `@NApiVersion` and `@NScriptType`.

JSDoc Tag	Possible Values	Required/Optional	Description
<code>@NApiVersion</code>	2.0 2.x 2.X	Required for entry point scripts Optional for custom modules	This JSDoc tag is used in two ways: <ul style="list-style-type: none"> ■ For SuiteScript 2.0 entry point scripts, this tag is a required declaration. It signifies to NetSuite the SuiteScript version to use. ■ For SuiteScript 2.0 custom modules that are not entry point scripts, this tag is an optional declaration. It can serve as a defense against future incompatible versions of SuiteScript (versions 3.x and higher) attempting to load it.
<code>@NModuleScope</code>	SameAccount TargetAccount Public	Optional	This JSDoc tag is used to control access to scripts and custom modules.

JSDoc Tag	Possible Values	Required/Optional	Description
			<ul style="list-style-type: none"> ▪ If the value is set to <code>SameAccount</code>, access to the module is limited to other modules from the same bundle, and modules native to the same source account or sandbox environment. Source code is not hidden at runtime. ▪ If the value is set to <code>TargetAccount</code>, access to the module is limited to other modules from the same bundle, and modules native to the same source account, target account, or sandbox environment. Source code is hidden at runtime. ▪ If the value is set to <code>Public</code>, any script in the account can load and use the module. Source code is hidden at runtime. <p>The default value is <code>SameAccount</code>. For more information, see Controlling Access to Scripts and Custom Modules.</p>
@NScriptType	BundleInstallationScript ClientScript MapReduceScript	Required for entry point scripts	This tag identifies the type of script defined in the file.

JSDoc Tag	Possible Values	Required/Optional	Description
	MassUpdateScript Portlet Restlet ScheduledScript Suitelet UserEventScript WorkflowActionScript		

Controlling Access to Scripts and Custom Modules

You can define the scope of environments (associated NetSuite accounts, sandboxes, and bundles) that may access a script. Access refers to whether the module can be loaded and invoked by another module. Modules are either loaded by the NetSuite system (via entry point modules) or by other modules (as libraries).



Important: Before deploying a SuiteApp, make sure that you review the module scope. This practice will help you to avoid:

Deploying a SuiteApp that doesn't work as expected because it can't access a necessary module in another bundle.

Providing wider access than desired to third parties and other accounts.

To set the scope for a script, you must include the appropriate value using the JSDoc tag. This tag is optional but recommended.

If you do not set the scope, `SameAccount` applies as the default. For information about adding a JSDoc tag, see [SuiteScript 2.0 JSDoc Validation](#).

The following table lists and describes the access control modifiers (supported module scopes) that are available in SuiteScript 2.0:

ModuleScope Value	Description	Access Disallowed When:	Visibility of Source Code	Examples
SameAccount	Limits script access to modules native to the same environment in which the script was created and uploaded. This environment includes the account from which a bundle is installed and can also include the related family of sandbox accounts. For more information about sandbox accounts, see the help topic	A bundled module from a different source account attempts to import the module.	Visible during runtime	<ul style="list-style-type: none"> ■ Installing a customization from a single sandbox to a production environment that is linked to the same account. ■ Distributing a bundle with private business logic as an ISV (Independent Software Vendor).

ModuleScope Value	Description	Access Disallowed When:	Visibility of Source Code	Examples
	Understanding NetSuite Account Types. A script file that is 'native' to the environment is added to an account using an authenticated user session.			
TargetAccount	Limits script access to modules native to the same source or target environment as the script. A source environment includes the NetSuite account (and any associated sandboxes) from which a bundle is installed into another account. A target environment includes the NetSuite account (and any associated sandboxes) into which a bundle is installed (whether via Bundle Copy or Bundle Install).	A bundled module that is not native to the target or source account attempts to import the module.	Hidden during runtime	<ul style="list-style-type: none"> ■ Account administrators distributing private business logic between accounts they control, and the modules in the bundle are needed by other modules created in the target account ■ Distributing a bundle with public APIs intended for import only by modules native to the target account.
Public	Any bundle (native and third party) that is installed in the account can run the script.	The script is not installed in the account.	Hidden during runtime	<ul style="list-style-type: none"> ■ Customers installing customizations from multiple sandbox accounts to production, where modules from different sandboxes need to load each other. ■ Installing a customization from a development account to a sandbox or production account. ■ Developing open source code.

For more information about packaging and distributing bundles (SuiteApps), see .

NetSuite Development Accounts and Module Scope



Warning: If you use a NetSuite Development account to work on a module that you intend to distribute or test, NetSuite recommends that you choose a module scope value rather than accept the default.

Keep in mind that development accounts are limited to 5 users and isolated from production and sandbox accounts. Set the value to `Public` if other bundles or accounts depend on using a module that is sourced from a development account.

If the default module scope is used on a script in a development account that is packaged into a bundle and installed to production or sandbox accounts, that script will not be accessible to modules installed from other NetSuite accounts. For example, when developers use multiple developer accounts to collaborate on a project, the `SameAccount` module scope might not be an appropriate access control level. This scope prevents modules installed from different accounts from loading one another.

For more information about development accounts, see the help topics [The Development Account](#), [NetSuite Development Accounts](#), and the [SuiteApp Development Process with SuiteBundler](#).

SuiteScript 2.0 Entry Point Script Deployment

After you write an entry point script, you must take the following steps to run the script in your NetSuite account:

1. Make sure that the file has the required script elements, as described in [SuiteScript 2.0 Entry Point Script Validation](#), and upload the file to your File Cabinet.
2. Do one of the following:
 - Deploy your script on one or more record types, as described in [SuiteScript 2.0 Record-Level Scripts](#).
 - If your script is a client script that should be deployed only at the form level, follow the steps described in [SuiteScript 2.0 Form-Level Scripts](#).

For help understanding the difference between deploying a client script at the record level versus the form level, see [Record-Level and Form-Level Scripts](#). Note that only client scripts can be deployed at the form level. All other types of entry point scripts can be deployed at the record level only.

Record-Level and Form-Level Scripts

A client script can be deployed in one of two ways: at the record level, or at the form level.

When you deploy a client script at the record level, you deploy it globally on one or more record types. The script runs on all forms associated with the record type. By contrast, with a form-level deployment, you attach the script to a custom form associated with a record type, and the script runs only when that form is used.

For example, you could use record-level deployment to configure a client script to run on the employee record type. With this approach, the script runs whenever the employee record is used, regardless of which form is being used. With record-level deployment, it is also possible to limit the script by configuring it to be available only to certain audiences. With this approach, the script runs only when the form is used by people in certain roles, groups, or other classifications, as configured on the Audience subtab of the script deployment record. However, with record deployment, you cannot limit the script to only one form. The script behaves the same way on all forms associated with the record type.

By contrast, you could attach the client script to only one custom entry form for the employee record type. If a user then opened an employee record using the standard entry form, the script would not run. You can attach a client script to any custom entry form, custom transaction form, or custom address form. However, you cannot limit the audience for a form-level script.

Note that only client scripts can be deployed at the form level. All other entry point scripts can be deployed at the record level only.

For full details about deploying a script at the record level, see [SuiteScript 2.0 Record-Level Scripts](#). For details about deploying a client script at the form level, see [SuiteScript 2.0 Form-Level Scripts](#).



Note: Record-level client scripts can also be used on forms and lists that have been generated through [UI Objects](#) during [Suitelets](#) development. Form-based client scripts cannot be used by Suitelets.

SuiteScript 2.0 Entry Point Script Validation

For a SuiteScript 2.0 entry point script to be usable, the script must contain certain required elements. The system parses your file for these elements when you upload the file to the File Cabinet. At that time, the system also saves data about your file. This data is used later when you create a script record based on the script.

If NetSuite detects that an element within the file is missing or formatted incorrectly, it returns an error. Errors can be returned at the time you upload the file to the File Cabinet, or when you try to create a script record. If you are attaching a client script to a custom form, errors can also be returned at that time.

For more details, see the following sections:

- [Entry Point Script Validation Guidelines](#)
- [Entry Point Script Validation Examples](#)
- [Entry Point Script Validation Error Reference](#)

Entry Point Script Validation Guidelines

For an entry point script to be properly formatted, its structure must meet certain guidelines. Additionally, the script must include two required JSDoc tags.

If the script does not meet all requirements, you cannot create a script record based on the script, nor can you attach it to a form. In some cases, you are not permitted to upload the file. Similarly, if a correctly formatted script has been attached to a script record or a custom form, you are not permitted to edit the file and save changes that would introduce errors.

For details, see the following sections:

- [Entry Point Script Validation Terms and Overview](#)
- [Correct Structure for Entry Point Scripts](#)
- [Required JSDoc Tags for Entry Point Scripts](#)

Entry Point Script Validation Terms and Overview

When learning about script validation requirements and errors, it is important to understand certain terms. These terms are described in the following illustration and table.

```


    /**
     * @NApiVersion 2.x
     * @NScriptType UserEventScript
     */
    define(['N/record'],
        function(record) {
            function beforeLoad(context) {
                if (context.type !== context.UserEventType.CREATE)
                    return;
                var customerRecord = context.newRecord;
                customerRecord.setValue('phone', '555-555-5555');
                if (!customerRecord.getValue('salesrep'))
                    customerRecord.setValue('salesrep', 46);
            }
            function beforeSubmit(context) {
                if (context.type !== context.UserEventType.CREATE)
                    return;
                var customerRecord = context.newRecord;
                customerRecord.setValue('comments', 'Please follow up with this customer!');
            }
            function afterSubmit(context) {
                if (context.type !== context.UserEventType.CREATE)
                    return;
                var customerRecord = context.newRecord;
                if (customerRecord.getValue('salesrep')) {
                    var call = record.create({
                        type: record.Type.PHONE_CALL,
                        isDynamic: true
                    });
                    call.setValue('title', '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('Call record created successfully', 'Id: ' + callId);
                    } catch (e) {
                        log.error(e.name);
                    }
                }
            }
        }

        return {
            beforeLoad: beforeLoad,
            beforeSubmit: beforeSubmit,
            afterSubmit: afterSubmit
        };
    });


```

Callout	Description
1	JSDoc tags
2	JSDoc tag values
3	entry point function definitions
4	interface (sometimes called script type interface)
5	entry points
6	entry point functions (sometimes called script type functions)

At a high level, the following rules apply:

- Each script must have a structure that meets certain criteria, as described in the next section, [Correct Structure for Entry Point Scripts](#).
- Each script file must include two JSDoc tags: @NApiVersion and @NScriptType. For more details, see [Required JSDoc Tags for Entry Point Scripts](#).

Correct Structure for Entry Point Scripts

An entry point script cannot be associated with a script record or custom form unless the script meets certain criteria. That is, all of the following must be true:

- The script must include one and only one properly structured define statement. The script cannot include multiple define statements.
- The define statement cannot include direct references to SuiteScript 2.0 API, objects or enums. All such references must be wrapped in a function.
- The script's return statement must include an interface whose entry points are associated with one and only one script type. Put another way, the return statement must include only one script type interface.
- The interface must include at least one entry point.
- Each entry point must correspond with an entry point function. All entry point functions must be defined in the same file.
- The script type interface implemented must match the @NScriptType value.
- JavaScript syntax errors are not permitted.

Required JSDoc Tags for Entry Point Scripts

Every entry point script must contain the JSDoc tags described in the following table. Note that this table includes details about some errors, but for full details about validation errors, see [Entry Point Script Validation Error Reference](#).

JSDoc Tag	Possible Values	Purpose	Possible Errors
@NApiVersion	2.0 2.x 2.X	Identifies the SuiteScript version to use.	If you omit this tag within a file that is properly formatted in all other ways for SuiteScript 2.0, the upload of the script fails. For details, see FAILED_TO_VALIDATE_SCRIPT_FILE . If you use an invalid value for this tag, the upload of the script fails. For details, see INVALID_API_VERSION .
@NScriptType	BundleInstallationscript ClientScript MapReduceScript MassUpdateScript Portlet Restlet ScheduledScript Suitelet UserEventScript WorkflowActionScript	Identifies the type of script defined in the file.	If you use an @NScriptType value that is incompatible with the entry points included in the script, the upload of the script fails. For details, see WRONG_SCRIPT_TYPE . If you omit this tag within a file that is properly formatted in all other ways for SuiteScript 2.0, you can upload the file, but you cannot create a script record for it, nor can you attach it to a form. For details, see FAILED_TO_VALIDATE_SCRIPT_FILE .

For full details about working with JSDoc tags in SuiteScript 2.0, including details on formatting them properly, see [SuiteScript 2.0 JSDoc Validation](#).

Entry Point Script Validation Examples

The following examples show correct and incorrect code snippets.

Correct: Includes All Required Elements

The following script is correctly formatted. It includes a define statement, a return statement with an entry point, and a function that corresponds with the entry point. Additionally, the value of the @NScriptType tag matches the script type interface used.

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

define(['N/record'],
    function(record)
    {
        function myBeforeSubmitFunction(context)
        {
            if (context.type !== context.UserEventTypes.CREATE)
                return;
            var customerRecord = context.newRecord;
            customerRecord.setValue({
                fieldId: 'comments',
                value: 'Please follow up with this customer!'
            });
        }
        return {
            beforeSubmit: myBeforeSubmitFunction
        };
    });
});
```

Incorrect: Missing Return Statement

The following script does not have a return statement. If you try to upload a script structured this way, the system returns an error reading “SuiteScript 2.0 entry point scripts must implement one script type function.” For details, see [SCRIPT_OF_API_VERSION_20 MUST](#)

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

define(['N/record'],
    function(record)
    {
        function myBeforeSubmitFunction(context)
        {
            if (context.type !== context.UserEventTypes.CREATE)
                return;
            var customerRecord = context.newRecord;
            customerRecord.setValue({
```

```

        fieldId: 'comments',
        value: 'Please follow up with this customer!'
    });
}
});

```

Incorrect: Missing Define Statement

The following script uses a require statement instead of a define statement. If you try to upload a script structured like this, the system returns an error reading “SuiteScript 2.0 entry point scripts must implement one script type function.” For details, see [SCRIPT_OF_API_VERSION_20 MUST ...](#)

```

/**
 * @NApiVersion 2.x
 * @NScriptType UserEventScript
 */
require(['N/record'],
    function(record)
{
    function myBeforeSubmitFunction(context)
    {
        if (context.type !== context.UserEventTypes.CREATE)
            return;
        var customerRecord = context.newRecord;
        customerRecord.setValue('comments',
            'Please follow up with this customer!');
    }
});

```

Incorrect: Missing JSDoc tag

The following script is incorrectly formatted because it does not have an @NScriptType tag. The system would permit you to upload a file with this script. However, if you try to create a script record with this script, the system returns an error reading “@NScriptType is mandatory for 2.0 entry point script.” For details, see [MISSING_SCRIPT_TYPE](#).

```

/**
 * @NApiVersion 2.0
 */
define([ 'N/record' ], function(record) {
    function myBeforeSubmitFunction(context) {
        if (context.type !== context.UserEventTypes.CREATE)
            return;
        var customerRecord = context.newRecord;
        customerRecord.setValue('comments',
            'Please follow up with this customer!');
    }
    return {
        beforeSubmit : myBeforeSubmitFunction
    };
});

```

Incorrect: Missing JSDoc tag

The following script is incorrectly formatted, because it does not have an @NApiVersion tag. If you try to upload a script structured like this, the system returns an error reading "Failed to validate script file." For details, see [FAILED_TO_VALIDATE_SCRIPT_FILE](#).

```
/**
 * @NScriptType userevents
 */

define([ 'N/record' ], function(record) {
    function myBeforeSubmitFunction(context) {
        if (context.type !== context.UserEventTypes.CREATE)
            return;
        var customerRecord = context.newRecord;
        customerRecord.setValue('comments',
            'Please follow up with this customer!');
    }
    return {
        beforeSubmit : myBeforeSubmitFunction
    };
});
```

Entry Point Script Validation Error Reference

The following table describes errors that can be returned when working with entry point scripts. These errors can be thrown during the process of uploading entry point scripts, when creating script records, or when attaching scripts to forms. Some errors can also be returned when editing script files that have already been uploaded to NetSuite, attached to script records, or attached to forms.

Error Code	Long Description
CANNOT_CHANGE_API_VERSION	This file is used by a SuiteScript {1} script; you cannot change the API version of the file.
FAILED_TO_VALIDATE_SCRIPT_FILE	Failed to validate script file: {1}
INVALID_API_VERSION	Invalid JSDoc tag value; valid values are: @NApiVersion [2.X, 2.x, 2.0]
INVALID_JSDOC_TAG_VALUE	Invalid JSDoc tag value; valid values are: {1}
MISSING_SCRIPT_TYPE	@NScriptType is mandatory for 2.0 entry point script.
MULTIPLE_DEFINE_CALLS	Invalid define call, define should only be called one time per module. Define calls found at the following line numbers: {1}
SCRIPT_OF_API_VERSION_20_CANNOT_IMPLEMENT_MORE_THAN_ONE_SCRIPT_TYPE_INTERFACES (See SCRIPT_OF_API_VERSION_20 CANNOT ...)	SuiteScript 2.0 entry point scripts cannot implement functions for more than one script type.
SCRIPT_OF_API_VERSION_20_MUST_IMPLEMENT_A_SCRIPT_TYPE_INTERFACE (See SCRIPT_OF_API_VERSION_20 MUST)	SuiteScript 2.0 entry point scripts must implement one script type function.

Error Code	Long Description
SS_V2_FILE_USED_FOR_FORM_SCRIPTS_MUST_IMPLEMENT_CLIENT_SCRIPT_TYPE (See SS_V2_FILE_USED_FOR_FORM_SCRIPT)	SuiteScript version 2 file used for form scripts must implement client script type
SYNTAX_ERROR	Syntax error: {1}
THE_FILE_IS_USED_AS_SCRIPT_TYPE_1_CANNOT_BE_CHANGED_TO_2 (See THE_FILE_IS_USED_AS_SCRIPT_TYPE_1 ...)	The file is used as script type {1}, cannot be changed to {2}
WRONG_SCRIPT_TYPE	Script file includes @NScriptType {1}; this script type cannot be used to {2}.

CANNOT_CHANGE_API_VERSION

The long description for this error code is: This file is used by a SuiteScript {1} script; you cannot change the API version of the file.

This error can be displayed when you attempt to edit a script file associated with a script record or a custom form. Specifically, this error is displayed if you try to edit and save changes to the @NApiVersion value. For example, if you try to remove the expression @NApiVersion 2.0 from a version 2.0 script, the system displays this error. Similarly, if you try to add @NApiVersion 2.0 expression to a version 1.0 script, the system displays this error.

In some cases, this error is preceded by the [INVALID_API_VERSION](#) error. The INVALID_API_VERSION error is included if the new @NApiVersion value you are trying to use is invalid.

Note that SuiteScript 1.0 files do not use the @NApiVersion tag. For help creating and uploading SuiteScript 1.0 files, see the help topic [Running Scripts in NetSuite Overview](#).

FAILED_TO_VALIDATE_SCRIPT_FILE

The long description for this error code is: Failed to validate script file: {1}

The system displays this error in a few cases. For example, you see this message if your script is formatted correctly for SuiteScript 2.0 in all ways except that it lacks the @NApiVersion tag.

INVALID_API_VERSION

The long description for this error code is: Invalid JSDoc tag value; valid values are: @NApiVersion [2.X, 2.x, 2.0]

This error is displayed when you attempt to upload a file with an invalid value for the @NApiVersion tag. It can also be displayed if you edit a script file that was already uploaded and try to modify the value of @NApiVersion tag.

If you see this error, check your file to make sure the @NApiVersion tag has a valid value. Valid values include the following:

- 2.0
- 2.x
- 2.X

INVALID_JSDOC_TAG_VALUE

The long description for this error code is: Invalid JSDoc tag value; valid values are: {1}

This error can be displayed if your script file uses an invalid value for any JSDoc tag. For example, if you use an invalid value for @NScriptType, the system displays this error.

To resolve the problem, check your file to make sure the value you used for the tag does not include typos or other errors.

MULTIPLE_DEFINE_CALLS

The long description for this error code is: Invalid define call, define should only be called one time per module. Define calls found at the following line numbers: {1}

This error is displayed if your script includes more than one define call. If you see this error, review the script and edit it appropriately.

MISSING_SCRIPT_TYPE

The long description for this error code is: @NScriptType is mandatory for 2.0 entry point script.

The system displays this error if you attempt to create a script record based on a SuiteScript 2.0 script that does not include the @NScriptType tag. To resolve the problem, edit the file and add the @NScriptType tag with the appropriate value.

The system does not display this error when you are uploading your file to the File Cabinet. It displays this error only when you try to create a script record, or if you edit a script file attached to a script record.

SCRIPT_OF_API_VERSION_20_CANNOT ...

The full code for this error is:

SCRIPT_OF_API_VERSION_20_CANNOT_IMPLEMENT_MORE_THAN_ONE_SCRIPT_TYPE_INTERFACES

The corresponding long description is: SuiteScript 2.0 entry point scripts cannot implement functions for more than one script type.

The system displays this error if your interface includes entry points from more than one script type. If you see this error, review your script's interface. Check that all entry points belong to a single script type. For details on the available entry points for each script type, see [SuiteScript 2.0 Script Types and Entry Points](#).

SCRIPT_OF_API_VERSION_20_MUST ...

The full code for this error is:

SCRIPT_OF_API_VERSION_20_MUST_IMPLEMENT_A_SCRIPT_TYPE_INTERFACE.

The corresponding long description is: SuiteScript 2.0 entry point scripts must implement one script type function.

This error signifies that your script is missing an interface or that an error exists within the interface. This error can be returned when you try to upload a file, or when you try to edit a file that was previously uploaded.

SS_V2_FILE_USED_FOR_FORM_SCRIPT

The full code for this error is:

SS_V2_FILE_USED_FOR_FORM_SCRIPTS_MUST_IMPLEMENT_CLIENT_SCRIPT_TYPE.

The corresponding long description is: SuiteScript version 2 file used for form scripts must implement client script type.

This error is displayed if you try to attach a script other than a client script to a custom form. Only a client script can be deployed at the form level. Other types of entry point scripts can be deployed only at the record level. For full details about working with form scripts, see [SuiteScript 2.0 Form-Level Scripts](#). For details about deploying other types of entry point scripts, see [SuiteScript 2.0 Record-Level Scripts](#).

SYNTAX_ERROR

The long description for this error code is: Syntax error: {1}

THE_FILE_IS_USED_AS_SCRIPT_TYPE_1 ...

The full error code for this error is:

`THE_FILE_IS_USED_AS_SCRIPT_TYPE_1_CANNOT_BE_CHANGED_TO_2`.

The corresponding long description is: The file is used as script type {1}, cannot be changed to {2}

You may see this error when editing a script file that is attached to an existing script record. If you edit the file so that it uses a different script type interface from what it previously used, and a different `@NScriptType` value, the system generates this error.

This error occurs because the Type field on the script record cannot be changed. To avoid this problem, make your changes in a file that is not already associated with a script record. Then create a new script record based on that file.

WRONG_SCRIPT_TYPE

The long description for this error code is: Script file includes `@NScriptType` {1}; this script type cannot be used to {2}.

The system displays this error if the file `@NScriptType` value is not compatible with the entry points in the script's return statement. If you see this error, double-check the `@NScriptType` value used in your script. Make sure that it corresponds with the entry points used by your script. If the tag's value does not match the script type interface, you cannot upload the file to the File Cabinet. For details on the available entry points for each script types, see [SuiteScript 2.0 Script Types and Entry Points](#).

SuiteScript 2.0 Record-Level Scripts

After you have written a valid entry point script, as described in [SuiteScript 2.0 Entry Point Script Validation](#), you must take the following steps if you want to deploy the script on records in your NetSuite account:

1. Use the script file to create a script record, as described in [Script Record Creation](#). The script record identifies the script's internal ID, whether or not the script is active, and other details.
2. Deploy the script, as described in [Script Deployment](#). You use script deployments to configure details regarding how and when the script runs. The types of choices you make vary depending on the script's type.



Important: These steps are specific to entry point scripts deployed at the record level.

Script Record Creation

After you have written an entry point script, you can create a script record for it. You must create a script record before you can deploy your script.

When you create a script record, you set some fields manually. By contrast, the system uses the data in the file to automatically set several key fields. These fields are described in the following table.

Field	Value Taken From
API Version	The @NApiVersion tag in your script file
Script File	The name of your script file.
Type	The @NScriptType tag in your script file
Functions (on the Script subtab)	The entry points defined within your script.

Note: Your script must be properly structured and annotated before it can be used to create a script record. For details, see [SuiteScript 2.0 Entry Point Script Validation](#).

At a high level, you can create a script record by using the following approach: Go to Customization > Scripting > Scripts > New. Select the appropriate file in the Script File dropdown list, then click **Create Script Record**.

In response, the system opens a new script record. The Type, API Version, and Script File fields are already populated. To complete the process, enter a name for the script record, and set any optional fields as needed. Then click **Save**.

After you save, the system shows the new script record in view mode. On the Scripts subtab, the system lists all possible entry point functions that could exist for this script type, with checks beside the ones used in your script.

The screenshot shows the 'Script' record view. At the top, there are buttons for Edit, Back, Deploy Script, Download XML, and Actions. Below these are sections for Type (User Event), Name (New User Event Script), Package, ID (customscript_usereventscript), and API Version (2.0). A 'DESCRIPTION' section shows Owner (John Smith) and Inactive status. The bottom navigation bar has tabs for Scripts (which is selected), Parameters, Unhandled Errors, Execution Log, Deployments, and System Notes. The Scripts tab displays a 'SCRIPT FILE' section with a preview of 'newUserEventScript.js', download and edit links, and three checkboxes for 'BEFORE LOAD FUNCTION', 'BEFORE SUBMIT FUNCTION', and 'AFTER SUBMIT FUNCTION'. All three checkboxes are checked.

The check boxes on the Scripts subtab cannot be edited directly. To check or clear the boxes, edit the script to add or remove the appropriate functions. When you save your changes, the script record's check boxes are updated to match the contents of the script file.

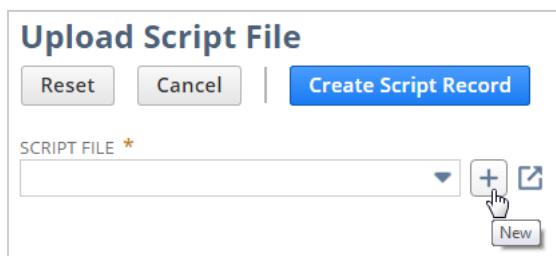
For more detailed help on creating a script record, see [Creating a Script Record](#).

Creating a Script Record

The process of creating a script record varies depending on the script's type. The following procedure uses general steps that apply to all types. For certain script types, additional steps may be required.

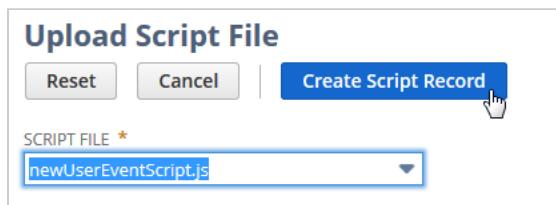
To create a script record:

1. Go to Customization > Scripting > Scripts > New.
- The system displays the **Upload Script File** page.
2. In the **Script File** dropdown list, select the appropriate SuiteScript 2.0 file. The dropdown list populates automatically with SuiteScript files that you have uploaded to your File Cabinet. This list includes version 1.0 and version 2.0 files.
- If the file you want to use has not been uploaded yet, you can upload it from this page. Point to the area at the right of the dropdown list to display a Plus icon.



Click the icon to display a dialog box for uploading a file from your local environment.

3. After you have populated the dropdown list, click the **Create Script Record** button.



The system displays the **Script** page, with your .js file listed on the **Scripts** subtab. The read-only **Type** field is automatically populated based on the content of your file. The read-only **API Version** field is automatically populated with the version number: **2.0**.

TYPE	User Event
NAME *	FollowUpEmail
ID	
API VERSION	2.0
Scripts Parameters Unhandled Errors Deployments	
SCRIPT FILE FollowUpEmail.js	
Save Cancel Reset	

Note: If the system redirects you to a page that prompts you to select a script type, that means that in Step 2 you identified a SuiteScript 1.0 file. You can click the Back button in your browser to return to the previous page and select a different file. Or, for details on creating a script record for SuiteScript 1.0, see the help topic [Steps for Creating a Script Record](#).

4. Enter a name in the **Name** field.
 5. In the **ID** field, optionally enter a custom ID for the script record. If the **ID** field is left blank, a system-generated internal ID is created for you when you save the record.
 6. If the **Portlet Type** field is displayed, select the appropriate value. This field is available only for the portlet script type. In these cases, it is a required field.
 7. In the **Description** field, optionally enter a description for the script.
 8. If appropriate, change the value of the **Owner** field. By default, this field is set to the currently logged-in user.
- After the script record is saved, only the owner of the record or a system administrator can modify the record.
9. If appropriate, check the **Inactive** box. When a script is set to Inactive, any deployments associated with the script are also inactive.
 10. On the **Parameters** subtab, define any parameters (custom fields) that are used by functions in the script.
 11. On the **Unhandled Errors** subtab, optionally define the people to be notified if script errors occur.

Three types of error notifications are sent:

- An initial email is sent about the first occurrence of an error within an hour.
- An email with aggregated error information for every hour is sent. (The error counter is reset when this email is sent.)
- An email is sent about the first 100 errors that have occurred after the error counter is set to 0.

For example, an error is thrown 130 times within an hour. An initial email is sent. After the 100th occurrence, another email is sent. Since there are an additional 30 occurrences within the same hour, a final summary email is sent at the end of the hour. During the second hour, if there are only 50 occurrences of the error, only one summary email is sent at the end of that hour.

Note: By default, the **Notify Script Owner** box is checked. Alternatively or in addition to the script owner, you can identify other people, as follows:

- Check the **Notify All Admins** box, if all administrators should be notified.
 - Select one or more groups from the **Groups** dropdown list. To define new groups, go to Lists > Relationships > Groups > New.
 - Enter the email addresses of any other users who should be notified. You can enter a comma-separated list of email addresses.
12. Optionally, define a deployment for the script record by clicking the **Deployments** subtab and adding a line to the sublist. For details on adding a line to this list, see [Deploying a Script by using the Deployments Sublist](#). If you want to add a deployment, another option is to skip the Deployments subtab and, in the last step in this procedure, click Save and Deploy.
 13. If this is a client script, optionally add lines to the **Buttons** sublist, which appears on the **Scripts** subtab.
 14. If the **Scripts** subtab includes the **Custom Plug-in Types** sublist, optionally add lines to this list.

15. To save the new record, click one of the following
 - **Save** – to save and close the record.
 - **Save and Deploy** – to save the script record and open a page that lets you create a new script deployment record.

Script Deployment

Before an entry point script will run in your NetSuite account, it must be deployed. You can deploy a script at the time you create a script record, or you can deploy it later. The deployment settings available vary depending on the script's type and on how you deploy the script.

When you deploy a script, the system creates a script deployment record. These records are listed at Customization > Scripting > Script Deployments. Deployments are also represented as lines on the Deployments subtab of the script record.

Multiple deployments can be created for the same script record. When multiple deployments exist, they are executed in the order in which they are listed on the Deployments sublist. This sequence typically corresponds with the order in which the deployments were created.

For more details, see the following topics:

- [Methods of Deploying a Script](#)
- [Deploying a Script by using the Deployments Sublist](#)
- [Updating a Script Deployment](#)
- [Managing Web Store Performance Impact](#)

Methods of Deploying a Script

To run an entry point script in your NetSuite account, you must create a script deployment for it. The deployment determines how and when the script runs. You can create a script deployment in any of the following ways. In most cases, you can also use these methods to edit a script deployment:

- [Work with the Deployments Sublist](#)
- [Use the Script Deployment Record Form](#)
- [Use N/record Module Methods](#)

Work with the Deployments Sublist

You can deploy a script at the same time as you create a script record. This process is described in [Deploying a Script by using the Deployments Sublist](#). This approach lets you define values for many deployment fields, although not all. You can use this technique at the following times:

- When you are creating a script record.
- For some script types, when you are editing an existing script record. However, for some script types, the Deployments sublist is read-only when you are editing the script record.

Use the Script Deployment Record Form

As an alternative, you can deploy a script by using the entry form for the script deployment record. You may want to use this approach for the following reasons:

- The script deployment record lets you access a greater number of fields than the Deployments sublist.

- If you have already created a script record, in some cases it is not possible to edit the Deployments sublist.

You can open the script deployment record entry form by using any of the following methods:

- View a script record, and click the **New Deployment** button.
- View an existing script deployment record, and select **Actions > New**.
- View the list of a script record's existing deployments, and click the **New Deployment** button. (To display the list of deployments, go to Customization > Scripting > Scripts, and click the Deployments link for the appropriate script record.)

Many of the body fields available during this process are the same as those you see when working with the Deployments sublist. For help with these fields, see [Deploying a Script by using the Deployments Sublist](#). For help with additional fields that are available, refer to the [SuiteScript Records Browser](#).

Use N/record Module Methods

You can create a deployment programmatically by using the `record.create(options)` method. When creating a script deployment record, set the `options.type` parameter to `record.Type.SCRIPT_DEPLOYMENT`. Similarly, if you want to modify a script deployment record programmatically, you can load it by using `record.load(options)`. You can modify the record by using other [N/record Module](#) methods.

For help with the field IDs available on the script deployment record, refer to the [SuiteScript Records Browser](#).

Deploying a Script by using the Deployments Sublist

If you want an entry point script to execute in NetSuite, you must deploy it.

This topic describes how to deploy a script by using the Deployments sublist. You can use this approach when you are in the process of creating a script record that you want to deploy. Depending on the script type, you can also use this approach when you are editing an existing script record.

 **Note:** For information on other ways of deploying a script, see [Methods of Deploying a Script](#).

Be aware that the script deployment process varies somewhat depending on the script type. The following procedure describes basic steps. For certain script types, additional steps may be required.

To deploy a script by using the Deployments sublist:

- If the script record is not already open for editing, open it. Go to Customization > Scripting > Scripts. Locate the script for which you want to create a script deployment. Click the corresponding **Edit** link.
- Click the **Deployments** subtab.
- Add a value to the sublist, as follows:
 - If the **Title** column is displayed, enter a title.
 - If the **Applies to** column is displayed, select the appropriate value in the dropdown list. Specifically, select the record type where you want to deploy the script. To deploy the script on all record types supported in SuiteScript, select **All Records**.
 - Enter a meaningful name in the **ID** column. The record's ID lets you work with the deployment programmatically. Note that the system automatically adds a prefix of **customdeploy** to the value you enter. If you do not specify an ID, a system-generated ID is created for you.

- Check or clear the **Deployed** box, as appropriate. For most script types, the default value is **Yes**. However, for map/reduce and scheduled scripts, the default is **Not Scheduled**.
- If the **Execute in Commerce Context** column is displayed, either leave the field cleared or click in the column to display a check box. This option is available for user event scripts only. It determines whether the script will be triggered by activity in the web store. For more details, see [Managing Web Store Performance Impact](#).
- In the **Status** column, select the appropriate deployment status. Note that the available values vary depending on the script type.
- If the **Event Type** dropdown list is displayed, optionally select a value from the dropdown list. This value identifies the event type that triggers the script execution.
- Optionally, in the **Log Level** field, specify which type of log messages will appear on the **Execution Log** tab when the script is executed.
- If the **Execute as Role** column appears, optionally select whether you want the script to execute using Administrator privileges, regardless of the permissions of the currently logged-in user.
- Click **Add** to add the new line to the sublist.

4. Click **Save**.

The system saves the deployment. If you view the Deployments subtab again, the deployment you just created is represented as a link. You can click the link to view the script deployment record that has just created been created. In some cases, this page lets you configure additional fields. For details, see [Updating a Script Deployment](#).

Updating a Script Deployment

If appropriate, you can edit an existing script deployment and make changes to it.

For some script types, you can make changes by editing the script record's Deployments sublist. However, for other types, the Deployments sublist is read-only. For these script types, you must open the full script deployment record for editing. You can do this programmatically, or by using the script deployment record entry form. This topic describes the latter method.

When you open the full script deployment record for editing, you have access to a greater number of fields than are available on the Deployments sublist. For example, the script deployment record includes the following subtabs:

- **Audience** subtab – When a script is deployed, it runs only in the accounts of the specified audience. This subtab lets you specify the roles that make up the script audience. In SuiteScript 2.0, this subtab is available for the following script types: client, portlet, RESTlet, Suitelet, and user event.
- **Links** subtab – For a Suitelet, lets you specify the menu paths that permit users to access the Suitelet.
- **Schedule** – for a map/reduce or scheduled script, lets you configure the schedule that determines when the script runs.



Important: If the script associated with the deployment is running in NetSuite, you cannot edit the deployment. You must wait until the script stops running.

To update a script deployment:

1. Go to Customization > Scripting > Script Deployments.
2. Locate the deployment you want to edit, and click the corresponding **Edit** link.
3. Make the appropriate changes.

4. Click **Save**.

Managing Web Store Performance Impact

When deploying a user event script, you can permit or prevent the deployment from being triggered by web store activity. You configure this behavior by using the script deployment record's **Execute in Ecommerce Context** option. This option lets you control whether the deployment is triggered by an event in SuiteCommerce Advanced, SuiteCommerce Site Builder, or SuiteCommerce InStore.

Scripts can significantly slow web store performance. By disassociating scripts from web store activity, you can improve web store response times.

You can set this option at the time you deploy a script. You can also go back and configure the option later, as described in the following procedure.

To enable or disable web store triggers for a user event script deployment:

1. Open the deployment for editing. For example, navigate to Customization > Scripting > Script Deployments. Locate the appropriate deployment and click the corresponding edit link.
2. On the script deployment record, locate the **Execute in Ecommerce Context**. Check or clear the box as appropriate.
3. Save the record.

i Note: Another option for optimizing web store performance is the **Asynchronous afterSubmit Sales Order Processing** feature. When you enable this feature, all afterSubmit user events and workflows triggered by web store checkout run asynchronously. For details, see the help topic [Enable Ecommerce Features](#).

Viewing System Notes

If appropriate, you can review details about the history of a script or script deployment record. These details identify the user who created the record, the context in which the record was created, and the date of the last change to the record.

To view a script deployment's history

1. Do one of the following:
 - To find a script, go to Customization > Scripting > Scripts.
 - To find a script deployment, go to Customization > Scripting > Script Deployments.
2. Locate the record you want to view, and click the corresponding **View** link.
3. Click the **System Notes** subtab.

i Note: In previous releases, details about these records were listed on its **History** subtab. However, that subtab is no longer updated. New activity is captured on the record's **System Notes** subtab.

SuiteScript 2.0 Form-Level Scripts

In some cases, you might want a client script to deploy on one form only. To configure this behavior, you attach the script to the form. You can attach a script to a custom entry form, a custom transaction form, or a custom address form.

With both custom entry forms and custom transaction forms, you can also use logic in the script to create a button or a menu item. These controls are sometimes referred to as custom actions. For custom address forms, you can deploy the script on the form, but you cannot configure custom actions.

The alternative to deploying your script on a form is to deploy it on a record type. In the latter case, the script is executed globally on that record type, regardless of which form is being used. By contrast, when you attach a client script to a form, it runs on that form only. For more details about the differences between these two approaches, see [Record-Level and Form-Level Scripts](#).

For details on deploying a client script at the form level, see the following sections:

- [Attaching a Client Script to a Form](#)
- [Configuring a Custom Action](#)



Important: Only client scripts can be attached to forms. Other types of entry point scripts can be deployed only at the record level. A script deployed at the record level is used on all forms associated with that record type. For help with record deployment, see [SuiteScript 2.0 Record-Level Scripts](#).

Attaching a Client Script to a Form

You can attach a client script to a form after you have made sure that it is a valid script, as described in [SuiteScript 2.0 Entry Point Script Validation](#).

After you attach a script to a form, it is deployed on that form and will run when the form is used. You can attach a client script to a custom entry form, a custom transaction form, or a custom address form.

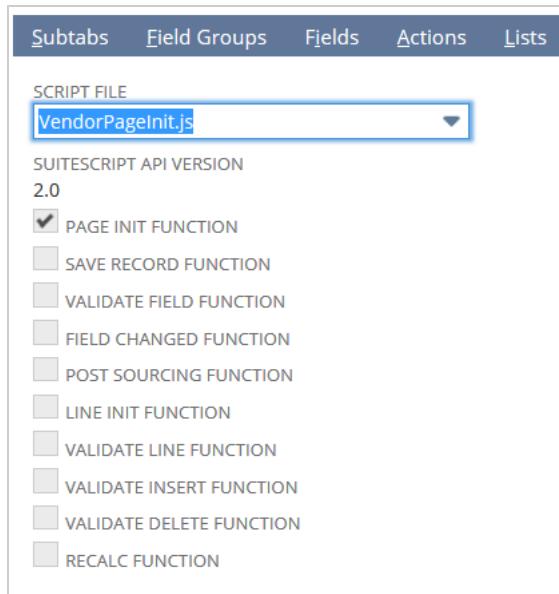


Important: Users must have at least the Edit level of the SuiteScript permission to attach a script to a custom form by editing the Custom Code tab of the form record. Users with the View level of the SuiteScript permission can see the Custom Code tab, but they cannot edit it.

To attach a client script to a form:

1. Navigate to the form to which you want to attach the script. For example, go to Customization > Forms > Entry Forms. Locate the appropriate form and click the corresponding **Edit** link.
2. Navigate to the **Custom Code** subtab.
3. In the **Script File** dropdown list, select the SuiteScript 2.0 script that you want to attach. If the file you want to use has not yet been uploaded to the File Cabinet, you can upload it from this page. To upload a file, point to the area at the right of the dropdown list to display a Plus icon. Click this icon to display a dialog box. You can use this dialog box to upload a file from your local environment.

After you populate the Script File field, the system updates the page to populate the **SuiteScript API Version** field. The page also updates to include a list of all possible client script entry point functions. Check marks are displayed next to the functions that are used by your script.



The check boxes on the **Custom Code** subtab cannot be edited directly. To check or clear the boxes, edit the script to add or remove functions as appropriate. When you save your changes, the script record's check boxes are updated to match the contents of the script file.



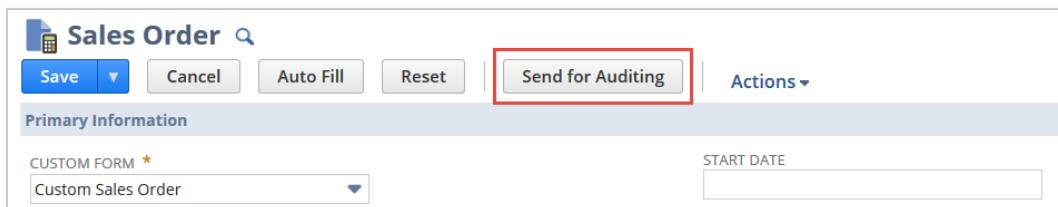
Note: If you edit the **Script File** field later and select a version 1.0 file instead of a version 2.0 file, the page reloads and includes fields into which you must manually enter the names of your entry-point functions. For details on this process, see the help topic [Step 3: Attach Script to Form](#).

4. Click **Save**.

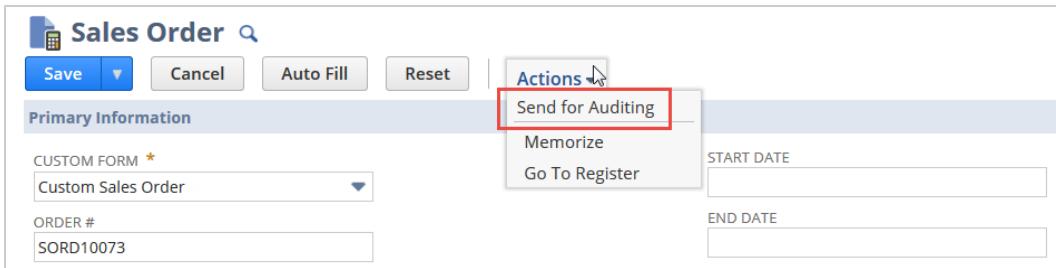
Configuring a Custom Action

In some cases, you might want to configure a custom action that uses logic contained in your client script. A custom action can be either of the following:

- A custom button that appears at the top of the page.



- A custom menu item that appears as an option when the user points to the Actions label.



You configure these elements by using logic contained in the client script attached to your form.



Note: It is not possible to configure a custom action for a custom address form. You can configure custom actions for custom entry and custom transaction forms.

To configure a custom action on a custom form:

1. If the form is not already open for editing, open it. For example, go to Customization > Forms > Transaction Forms. Locate the appropriate form and click the corresponding **Edit** link.
2. Navigate to the **Actions** subtab.
3. Navigate to the **Custom Actions** subtab.
4. In the **Label** column, enter a label for your button or menu item.
5. In the **Function** column, enter the name of the appropriate entry point function from your client script.
6. In the **Display as** column, select **Button** or **Menu** as appropriate.
7. Click **Save**.



Note: For more information about both custom actions and standard actions, see the help topic [Configuring Buttons and Actions](#). For help understanding the validation terms used in this topic, see [Entry Point Script Validation Guidelines](#).

SuiteScript 2.0 Custom Modules

- [Custom Modules Overview](#)
- [Writing a Custom Module](#)
- [Preparing to Add a Custom Module](#)
- [Naming a Custom Module](#)
- [Custom Module Examples](#)
- [Troubleshooting Errors](#)
- [Frequently Asked Questions: Custom Modules](#)

Custom Modules Overview

With SuiteScript 2.0, you have the ability to create custom modules (including third-party, AMD, and non-AMD modules). This supports modular and well-structured organization of code that is easier to read and modify. It also lets you build and access additional, customized API functionality.

Build a custom module to help you do the following:

- Group reusable functions into a common library file. Supporting code (such as helper functions and custom APIs) can be organized into reusable sections. These custom modules are loaded within your entry point script.
- Add custom modules to SuiteApps and expose those modules to third parties.
- Import a third-party API.
- Organize and separate utilities from business logic.

Writing a Custom Module

From the source `suitescript` file that defines your custom module, you will need to use the [define Function](#). Then, to load and use the module, specify the dependency with a [require Function](#) from an entry point script. You must also define a function that instantiates the module.

If desired, you can add standard and custom JSDoc tags to your custom module scripts. JSDoc tags are optional (unless the script is an entry point script). However, NetSuite recommends using the `@NApiVersion` and `@NModuleScope` tags in your custom modules. The SuiteScript API version tag can help protect against loading from future incompatible versions of SuiteScript (versions 3.x and higher). The module scope tag determines whether other scripts and accounts can access the custom module.

For custom JSDoc tags, SuiteScript 2.0 users can use JSDoc to create their own documentation for scripts, custom modules, and SuiteApps. To take advantage of this tool, developers must download JSDoc from the official website. For additional information on JSDoc, see <http://usejsdoc.org/>.

For script samples, see [Custom Module Examples](#).

To learn about setting up a custom module name, see [Naming a Custom Module](#).

Preparing to Add a Custom Module

Before you work on a custom module, make sure you're familiar with the following topics:

- [SuiteScript 2.0 – Script Architecture](#)
- [SuiteScript 2.0 Entry Point Script Validation](#)

- SuiteScript 2.0 Global Objects and Methods and [Module Dependency Paths](#)
- Controlling Access to Scripts and Custom Modules

Naming a Custom Module

Loading a custom module by name

You can set up a custom module name to aid re-use. After you configure a module name, you can require it without knowing the path.

Custom module loading by name also enables better interoperability with third-party libraries and may offer some protection against naming conflicts.

You will need to call `define(id, [dependencies,] callback)` and configure a [require Function](#).

The following steps demonstrate the steps to load a custom module by name:

1. Create your module file. For example `math.js`.
2. Author your custom module name and its contents:
3. To load the module by name, specify the module name (alias) and its path by configuring the `paths` parameter for a [require Function](#). See [require Configuration](#) for more information.

```
...
require.config({
  paths: {
    'math': 'SuiteScripts/Shim Example/math'
  },
...
}
```

4. Load the custom module in your SuiteScript 2.0 script by passing the name to the [define Function](#).

```
...
define(['math'],
  function (math) {
    return {
      beforeLoad: function beforeLoad() {
        log.debug('test', math.add(1,2));
      }
    });
...
}
```

For a code sample that shows configuration of the `paths` parameter, see [Example: Import a third-party JavaScript library](#).

Custom Module Examples

The following examples demonstrate using the `define` and `require` functions when working with custom modules.

- [Example: Define a custom utility module](#)

- Example: Import the custom utility module
- Example: Define a custom module for a SuiteApp
- Example: Import a third-party JavaScript library

Example: Define a custom utility module

The following file holds the definition of a custom module with a custom API. For example, to allow multiple scripts that depend on an incremental counter utility to require and import this functionality.

To protect against version incompatibility, this script includes the @NApiVersion tag.

```
/**  
 * counter.js  
 * @NApiVersion 2.x  
 */  
define(function(){  
    var counter = 0;  
  
    function incrementValue() {  
        counter++;  
    }  
    function getValue() {  
        return counter;  
    }  
  
    return {  
        increment: incrementValue,  
        value: getValue  
    }  
});
```

Example: Import the custom utility module

This example uses the [define Function](#) to include a native SuiteScript module and a custom module.

The module's file path is used to pass in the custom utility as a dependency. As a best practice, it does not include the .JS file extension.

```
/**  
 * customRecord.js  
 * @NApiVersion 2.x  
 */  
define(['N/record','./counter'],  
    function(record,counter){  
        function createCustomRec() {  
            record.create( );  
            counter.increment();  
        }  
  
        return {  
            createCustomRecord: createCustomRec  
        }  
   });
```

Example: Define a custom module for a SuiteApp

To define a bundled custom module that can be exposed to third parties, you must add the `@NModuleScope` JSDoc tag and assign it the value `public`.

```
/**  
 * @NApiVersion 2.0  
 * @NModuleScope public  
 */
```

Next, use the [define Function](#) so that your custom module is recognized as an AMD module. To use a module that is bundled within an external SuiteApp, you need to pass the bundle's file path, containing a valid bundle ID, within the `define()` function as follows:

```
define(['./bundle/<bundle ID>/<module path>'],  
    function(<module name>){  
        <logic goes here>  
    }  
);
```

If the bundle ID changes, the module loader will still attempt to locate it. The module loader uses the bundle ID to search the SuiteApp's deprecation path using valid [Bundle Virtual Paths](#).

Example: Import a third-party JavaScript library

- Import a third-party library
- Example: Add a non-AMD library

Import a third-party library

Some third-party libraries register as AMD compatible.

In that case, you can specify a require configuration that sets up the path where the module is found.

```
/**  
 * @NApiVersion 2.x  
 * @NScriptType userevents script  
 */  
require.config({  
    paths: {  
        "coolthing": "/SuiteScripts/myFavoriteJsLibrary"  
    }  
});  
define(['coolthing'],  
    function (coolthing)  
    {  
  
        return {  
            beforeLoad: function beforeLoad(ctx)  
            {  
                coolthing.times(2, function () {  
                    log.debug('log', 'log');  
                });  
            }  
        }  
    }  
);
```

```
    };
});

});
```

To use a version of jQuery, you can form your `require.config()` function as follows:

```
...
require.config({
    paths: {
        'jquery': '<jquery file path>'
    }
});
define(['jquery'], function(jquery) {
});
...
```

Note: If you want to use jQuery with a Suitelet, import it via an ad-hoc client script that is attached to the Suitelet using `Form.clientScriptFileId` or `Form.clientScriptModulePath`.

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.

Example: Add a non-AMD library

Before you can use non-AMD libraries in your SuiteScript 2.0 script, you must configure your script to load the dependencies for these libraries. You do this by configuring a `require()` function.

In this example, the following file would be uploaded to the file cabinet (as a JavaScript file that does not register as AMD compatible).

```
var myMath = {
    add: function(num1, num2) {
        return num1 + num2;
    },
    subtract: function(num1, num2) {
        return num1 - num2;
    },
    multiply: function(num1, num2) {
        return num1 * num2;
    },
    divide: function(num1, num2) {
        return num1 / num2;
    },
}
```

If a library does not register as an AMD module, you need to use the `shim` parameter.

The module can be loaded as a dependency in an entry point script, like the following user event script.

```
/**
 * @NApiVersion 2.x
 * @NScriptType usereventscript
 */
require.config({
```

```

paths: {
    'math': 'SuiteScripts/Shim Example/math'
},

shim: {
    'math': {
        exports: 'myMath'
    },
}

});

define(['math'],
    function (math) {
        return {
            beforeLoad: function beforeLoad() {
                log.debug('test', math.add(1,2));
            }
        }
    });
}
);

```

In the previous sample, `require.config()` returns a new require object when you pass in a `context` property. That new require is configured with the configuration object passed in.

This is useful if you want to have different configurations for some of the dependency modules used.

You need to require the desired module, like “math”, with the new require object. Otherwise, the new require configuration is not used when getting the “math” module.

Troubleshooting Errors

Module not found

You may see this error if you are using the `require` function. The `require` function has no global context. Consequently, relative paths do not work for the `require` function.

If you receive the Module does not exist error, try replacing any relative paths with an absolute path.

For more information, see [require Function](#).

You do not have permission to load this module.

Review your module scope settings. For a full description of support module scopes, see [Controlling Access to Scripts and Custom Modules](#).

Frequently Asked Questions: Custom Modules

When should I use `require` versus `define`?

Always use the [define Function](#) in entry point scripts and when creating new modules. Use the [require Function](#) for loading and using existing modules.

There is a performance advantage to using the require function. Generally, give preference to using the require Function whenever you can. The define Function will imports all dependencies. The require function loads dependencies only as they are needed.

Here is a summary of when to use or not use these functions:

Function	Example	Caveat
define	To define entry point scripts To create new modules. To export modules	If you use a define function, you do not receive the performance benefit of progressive loading.
require	To import modules via relative path To step through code in the SuiteScript debugger To support building a test framework For progressive loading of dependencies	If you use a require function, you may need to import a define that provides correct context.

For more information, see [SuiteScript 2.0 – Script Architecture](#).

Are third-party libraries supported?

Yes. You can define the library as a custom module, set up a global identifier to reference it, and configure it as a dependency. See [Import a third-party library](#) and [Example: Import a third-party JavaScript library](#).

SuiteScript 2.0 Scripting Records and Subrecords

For information on using SuiteScript 2.0 to work with records and subrecords, see the following topics:

- [Record and Subrecord Scripting Overview](#)
- [Scripting Records](#)
- [Scripting Subrecords](#)

Record and Subrecord Scripting Overview



Important: This topic is a work in progress. Additional updates are forthcoming.

Scripting Records



Important: This topic is a work in progress. Additional updates are forthcoming.

Scripting Subrecords

For details on using SuiteScript 2.0 to work with subrecords, see the following sections:

- [Understanding Subrecords](#)
- [Subrecord Scripting in SuiteScript 2.0 Compared With 1.0](#)
- [Scripting Subrecords that Occur on Sublist Lines](#)
- [Scripting Subrecords that Occur in Body Fields](#)

Understanding Subrecords

Like a record, a subrecord represents a way of storing data.

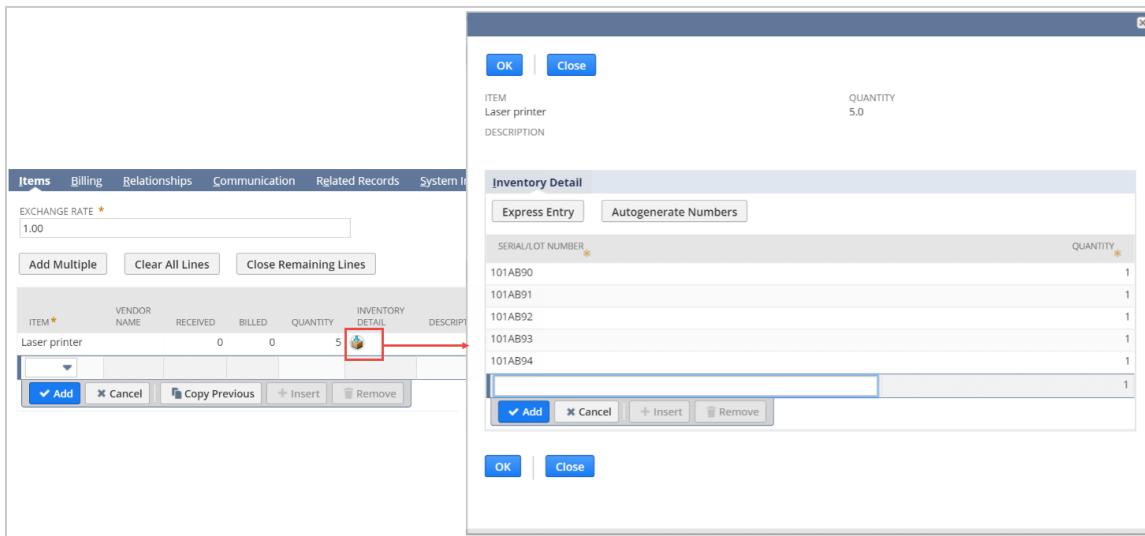
In some ways, subrecords are similar to records. For example, as with records, there are multiple types of subrecords, each with a different purpose and a different set of available fields.

However, an instance of a subrecord can exist **only** to hold information relevant to a particular instance of a record, which is known as the subrecord's parent. A subrecord cannot exist independently. It also cannot be accessed outside the context of its parent record. For example:

- An employee record may list several addresses for the employee. Each address is stored in an instance of the address subrecord. Each subrecord can be accessed from a line on the employee record's Address sublist.
- A blanket purchase order may authorize the purchase of a large quantity of a particular item, with that quantity to be divided up and tracked by individual purchase orders. The timing for the creation of each individual purchase order can be determined by dates entered in an order schedule subrecord. The subrecord can be accessed from a body field on the blanket purchase order.

- An assembly build record may exist to track the assembly of a large quantity of a serialized assembly item. Depending on the features enabled in your account, the assembly build record may include the serial numbers of each assembly to be completed. These serial numbers are listed in an inventory detail subrecord, which can be accessed from a body field on the assembly build record.

To open a subrecord in the UI, typically you click an icon on the record. The subrecord form opens in a separate window.



See the following sections for more details about subrecords:

- Supported Deployments for Scripts that Interact with Subrecords
- Body Field Subrecords and Sublist Subrecords
- Structure of a Subrecord
- Finding Subrecord Details in the Records Browser
- Understanding the Address Subrecord

Supported Deployments for Scripts that Interact with Subrecords

A script can interact with subrecord instances only if it uses a supported deployment.

To understand supported deployments, it's important to understand that every subrecord has a parent record. For more details on this relationship, see [Understanding Subrecords](#).

The following types of deployments are supported:

- Server-side Scripts Deployed on Parent Records
- Client Scripts Deployed on Parent Records (with Limitations)
- Client Scripts Deployed on Custom Address Forms

Server-side Scripts Deployed on Parent Records

If you want to create a server-side script that interacts with a subrecord, you can deploy the script on the parent record type. Alternatively, you can deploy the script on a different record type — but then

use the script to load the parent record. After loading the parent record, you can interact with the subrecord in the context of its parent.

Subrecord methods are not supported in beforeLoad user event scripts, except if the script creates or loads another record that is a parent, and interacts with the subrecord in the context of that parent.

You cannot deploy a server-side script directly to a subrecord type.

Client Scripts Deployed on Parent Records (with Limitations)

A client script may not create subrecords on the current record and is limited to read-only access of existing subrecords on the current record. The client script may remove the subrecord from the current record.

You cannot deploy a client script directly on a subrecord type. However, you can customize an address form, as described in the following section.

You cannot deploy a client script directly to a subrecord type.

Client Scripts Deployed on Custom Address Forms

If appropriate, you can create custom forms for the address subrecord. This process is described in [Customizing Address Forms](#). When working with a custom address form, you can attach a client script to the form by using the Custom Code subtab. In these types of scripts, you can interact with the subrecord using the same methods as you would with a record. This process is not covered in this chapter.

Body Field Subrecords and Sublist Subrecords

When a subrecord occurs on a record, it is always represented by a single field on the record. In the [SuiteScript Records Browser](#), this field is always listed as a field of type summary.

A field that contains a subrecord can exist either as a body or sublist field. For example, the subsidiary record has a body field called mainaddress, which stores an address subrecord. By contrast, the employee record permits the creation of multiple addresses, and each address is described in a sublist line. The sublist includes a field that contains the address subrecord instance.

This difference in where the summary field is placed affects the way you instantiate the subrecord.

For an example of each type of placement, see the following sections:

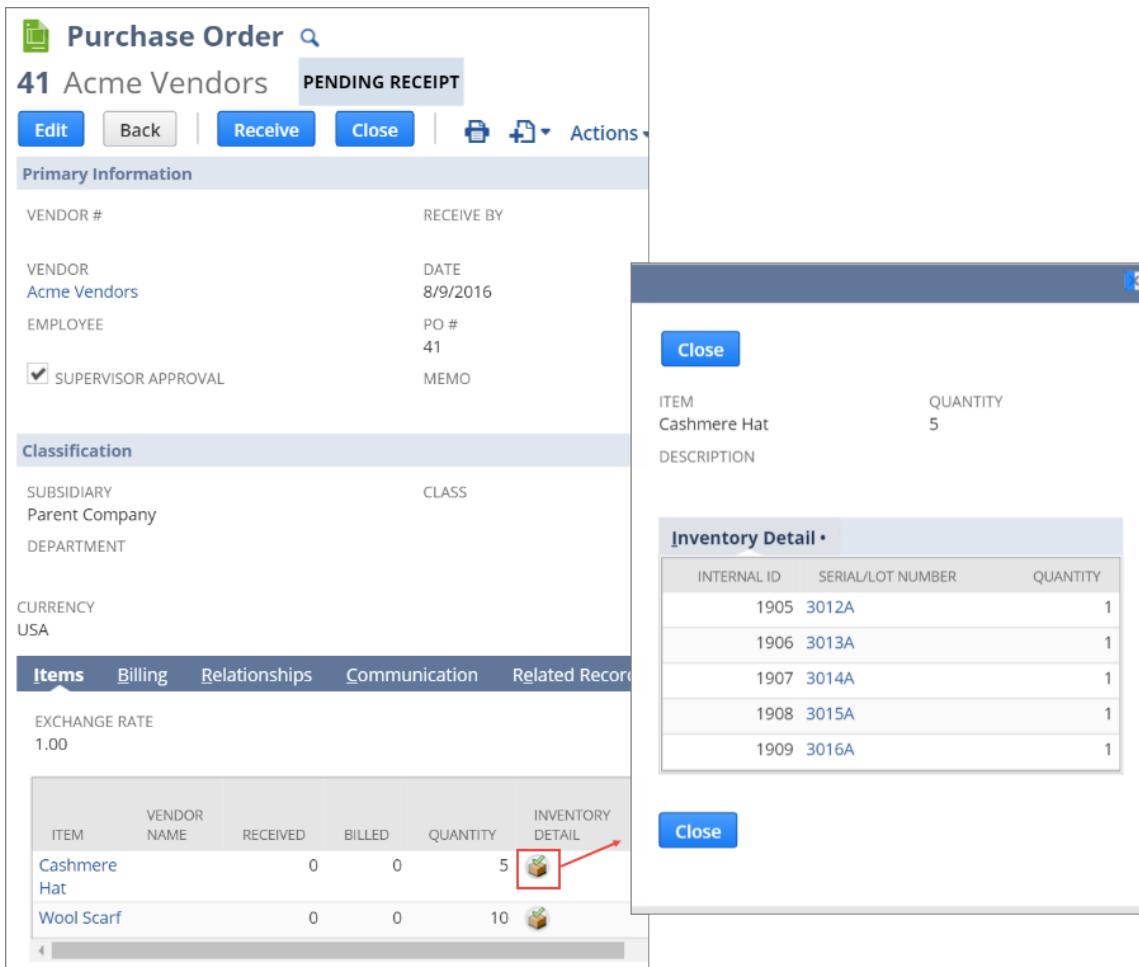
- [Example: Subrecord that Occurs in a Sublist Field](#)
- [Example: Subrecord that Occurs in a Body Field](#)

Example: Subrecord that Occurs in a Sublist Field

Many subrecord types can occur in a sublist field.

For example, depending on the features enabled in your account, the item sublist of a purchase order record can include an Inventory Detail column. If the item on the line is a serialized or lot-numbered item, you can create a subrecord instance in this column.

In the UI, you can view and set values in this subrecord by clicking the icon in the Inventory Detail column. Clicking this icon opens a new window that represents the subrecord.



Be aware that other fields on the sublist line are not part of the subrecord. For example, in the preceding screenshot, the values in the Item, Vendor Name, Received, Billed, and Quantity columns are not part of the subrecord.

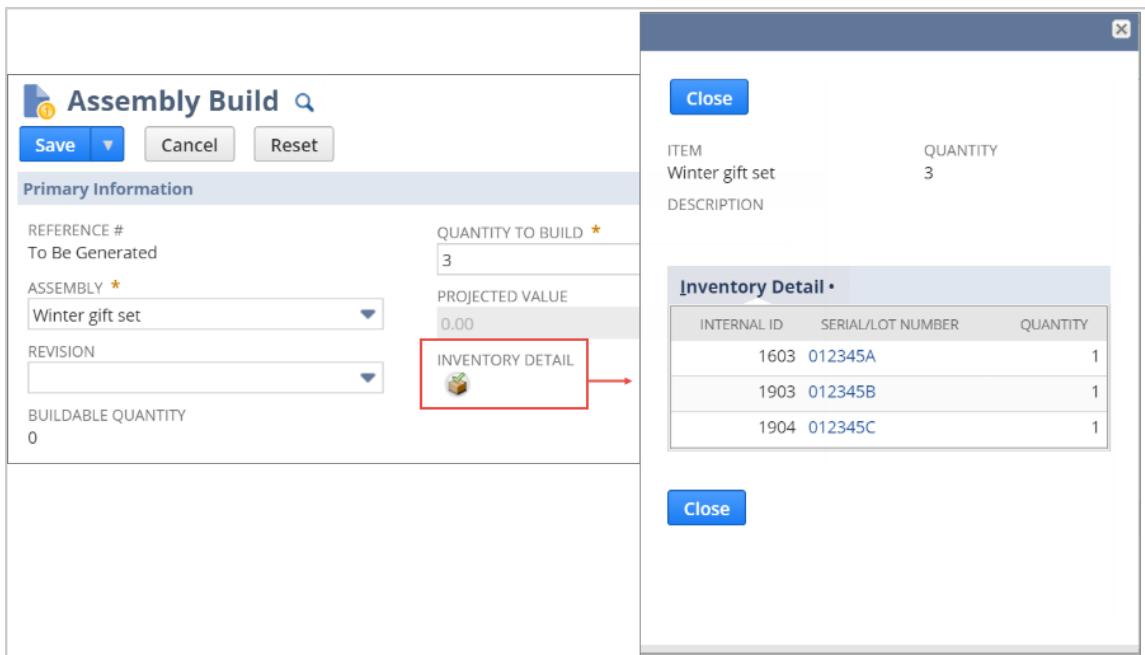
For details on using SuiteScript 2.0 to work with subrecords that exist on sublist lines, see [Scripting Subrecords that Occur on Sublist Lines](#).

Example: Subrecord that Occurs in a Body Field

A subrecord can also occur in a body field. In fact, the same type of subrecord that appears as a sublist field on one record type can appear as a body field on another record type.

For example, depending on the configuration of your account, the assembly build record can include an Inventory Detail body field. If the item in the record's Assembly field is a serialized or lot-numbered inventory item, you can create an Inventory Detail subrecord in the Inventory Detail field.

In the UI, you can view and set values in this subrecord by clicking the icon under the Inventory Detail label and opening a new window.



For details on using SuiteScript 2.0 to work with subrecords that exist in body fields, see [Scripting Subrecords that Occur in Body Fields](#).

Structure of a Subrecord

The fields available in a subrecord vary depending on the subrecord's type. A subrecord can have body fields, a sublist, or both.

When working with a subrecord's fields, be aware of the following:

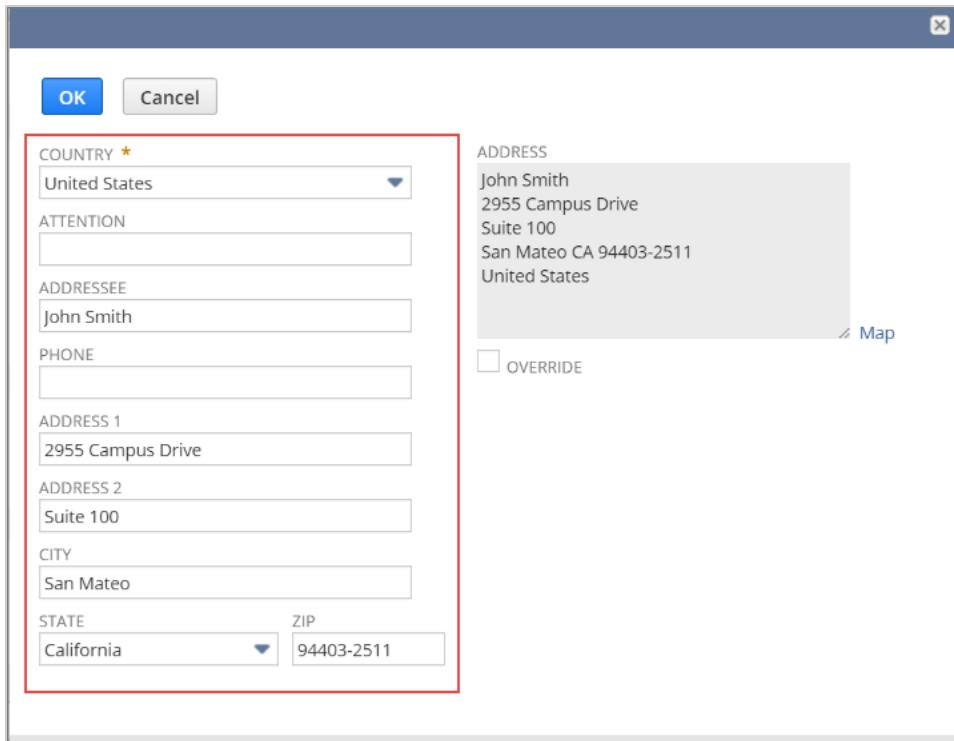
- After you instantiate a subrecord, you can set values on the subrecord's body and sublist fields by using the same methods as you would on a record.
- In general, the sublists that exist within subrecords are not labeled in the UI. To find the name of a subrecord's sublist, refer to the [SuiteScript Records Browser](#). For details on using the Records Browser to find details on subrecords, see [Finding Subrecord Details in the Records Browser](#).

For examples of subrecords that are structured in different ways, see the following sections:

- [Example: Writable Body Fields](#)
- [Example: Writable Sublist](#)
- [Example: Writable Body Fields and Sublist](#)

Example: Writable Body Fields

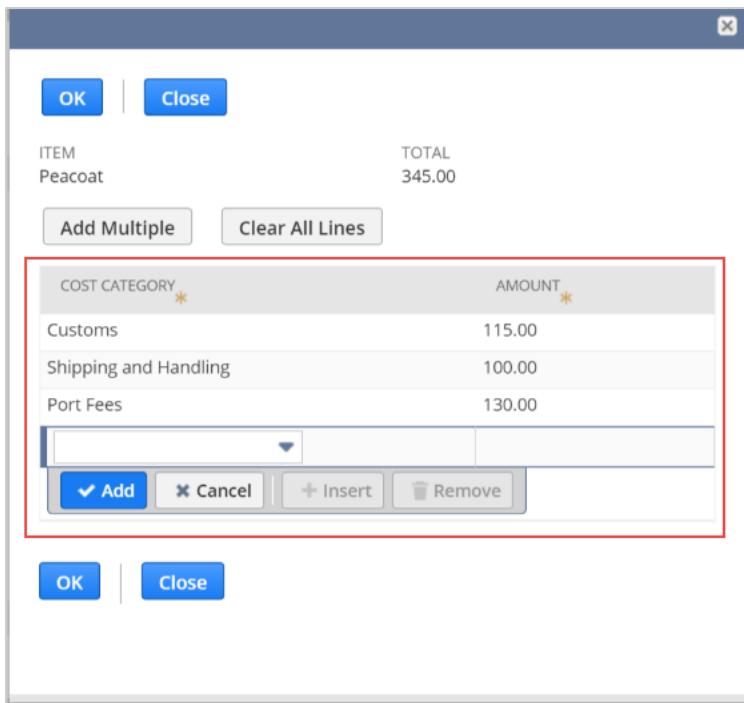
The address subrecord has several writable body fields, such as city, state, and zip. It has no sublist.



After you instantiate an address subrecord, you can set values for its body fields by using the `setValue()` method, the same as if you were setting values on an instance of a record. For more details, see [Example: Creating an Address Sublist Subrecord](#) and [Example: Creating an Address on a Subsidiary Record](#).

Example: Writable Sublist

The landed cost subrecord has a sublist that lets you list individual expenses associated with merchandise you have received. The body fields on this subrecord are read-only, but the sublist is writable. To add details about an expense, you add a line to the sublist.

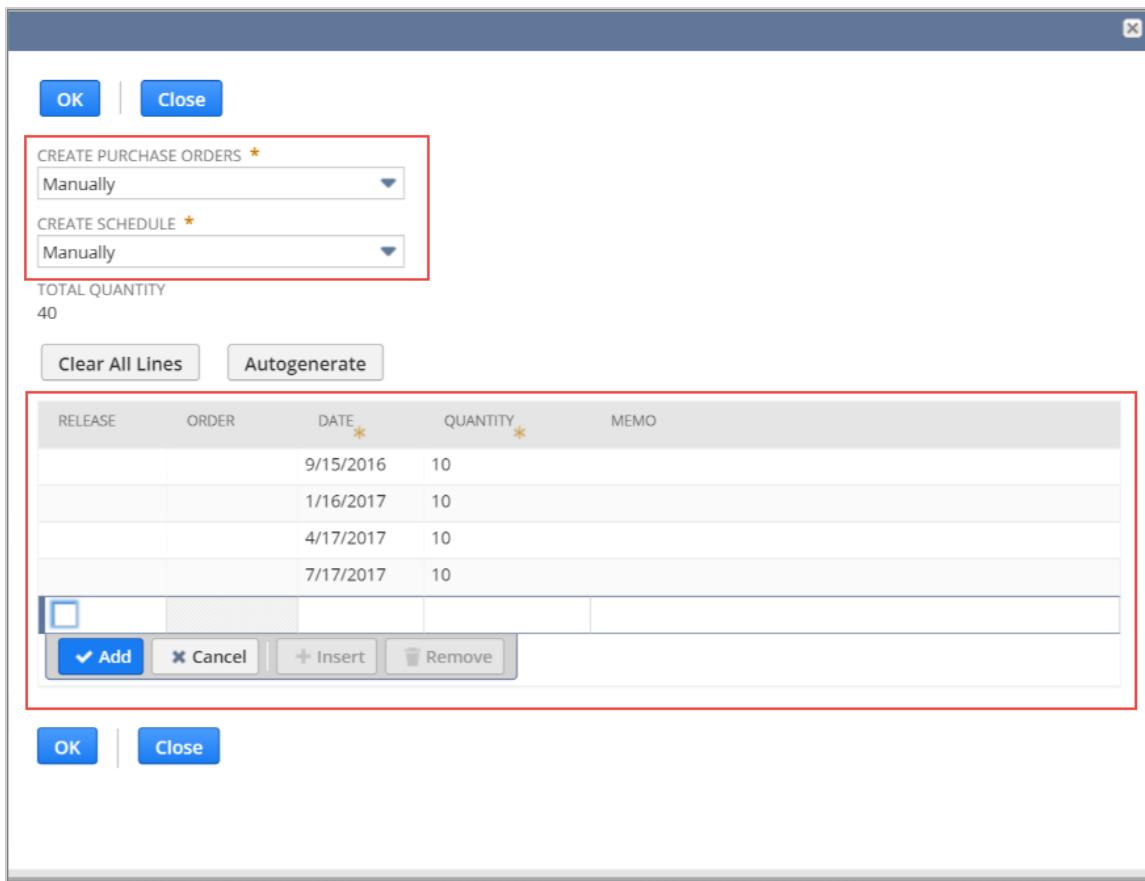


After you instantiate a landed cost subrecord, you can set values for its sublist fields by using the same methods you would to set values on a record's sublist: `setSublistValue()` and `setCurrentSublistValue()`.

For more details, see [Example: Creating a Landed Cost Sublist Subrecord](#).

Example: Writable Body Fields and Sublist

The order schedule subrecord has a sublist that lets you configure how and when upcoming purchase orders are to be created. This subrecord has body fields that let you specify various qualities of the schedule, such as whether individual purchase orders must be created manually. It also has a sublist that lets you enter dates for the upcoming purchase orders.



Again, after you instantiate an order schedule subrecord, you can set values its fields by using the same methods you would use to set values on a record. For body fields, use `setValue()`. For sublist fields, use `setCurrentSublistValue()` or `setSublistValue()`.

For more details, see [Example: Creating an Order Schedule Sublist Subrecord](#).

Finding Subrecord Details in the Records Browser

As with records, you can find important information about subrecords in the [SuiteScript Records Browser](#). The browser includes details about each subrecord type. It also includes information about record types that can be parents to subrecords. For more guidance, see the following sections:

- [Record Types that Can Be Parents](#)
- [Subrecord Types](#)

Record Types that Can Be Parents

Every subrecord instance must have a parent record. An instance of a subrecord exists only to provide information about an instance of a record.

The parent record includes a field that contains, or references, the subrecord. To create a subrecord instance on a record, you must reference this field. These fields are always identified in the Records Browser as fields of type summary.

item - Item			
Internal ID	Type	Label	Required
amount	currency	Amount	false
billvariancestatus	text		false
catchupperiod	select		false
class	select	Class	false
deferrevrec	checkbox		false
department	select	Department	false
description	textarea	Description	false
id	text		false
isvsoebundle	text		false
item	select	Item	true
itemsubtype	text		false
itemtype	text		false
line	text		false
linenumber	integer		false
location	select	Location	false
matrixtype	text		false
options	text		false
orderschedule	summary	Schedule	false
quantity	float	Quantity	false
quantityordered	float	Quantity Ordered	false
rate	rate	Rate	false
rateschedule	text		false
units	select	Units	false
vendorname	text	Vendor Name	false

i Note: Summary fields can occur either on the body of the parent record or in a sublist. For an overview of this distinction, see [Body Field Subrecords and Sublist Subrecords](#).

In some cases, other fields on a record can affect the availability or behavior of the summary field. For example:

- On an assembly build record, the availability of the inventorydetail summary field varies depending on the value of the record's item field. The summary field is available only if the item field references a serialized or lot-numbered assembly item. For an example of working with this record-subrecord combination, see [Example: Creating an Inventory Detail Subrecord on a Body Field](#).
- On a vendor bill record, the landedcost summary field is available only if the landedcostperline body field is set to true. For an example of working with this record-subrecord combination, see [Example: Creating a Landed Cost Sublist Subrecord](#).
- On a sales transaction, the value of the shippingaddress summary field is affected by the shipaddresslist field. For general details about shipping and billing addresses, see [Scripting Transaction Shipping and Billing Addresses](#). For a script sample, see [Example: Using SuiteScript 2.0 to Create a New Shipping Address](#).

Subrecord Types

As with record types, the Records Browser includes a reference page for each subrecord type. Subrecords are listed alphabetically with records.

Each subrecord is identified by a label displayed beneath the internal ID.

Order Schedule				
Internal ID: orderschedule				
Subrecord				
Fields				
Internal ID	Type	nlapiSubmitField	Label	Required
createpurchaseorder	select	false	Create Purchase Orders	true
createschedule	select	false	Create Schedule	true
currencyprecision	integer	false		false
enddate	date	false	End Date	false
externalid	text	false	ExternalId	false
item	integer	false		false
releasefrequency	select	false	Release Frequency	false
startdate	date	false	Start Date	false
total	float	false	Total Quantity	false

Sublists				
schedule - Schedule				
Internal ID	Type	Label	Required	
amount	poscurrency		false	
id	integer		false	
memo	text	Memo	false	
orderschedule	integer		false	
purchaseorder	select	Order	false	
quantity	posfloat	Quantity	true	
release	checkbox	Release	false	
trandate	date	Date	true	

As with record types, the reference page shows the subrecord's scriptable fields and sublists.

Note that in the UI, the sublists of subrecords typically are not labeled. However, the Records Browser displays the name of each sublist.

Understanding the Address Subrecord

The address subrecord has certain qualities that are unique. These characteristics can make the process of interacting with the address subrecord different from other subrecords. For details, see the following sections:

- [Billing and Shipping Addresses Can Be Sourced from Other Records](#)
- [The Address Subrecord Can Have Custom Forms](#)
- [Address Data Is Summarized on the Parent Record](#)



Note: See also [Scripting Transaction Shipping and Billing Addresses](#).

Billing and Shipping Addresses Can Be Sourced from Other Records

With most subrecord types, an instance of the subrecord is unique to the record where it was created. However, in some cases, a single address subrecord instance can be referenced by multiple records.

For example: You can create multiple addresses for an entity. If you later create a transaction for that entity, you can use one of the addresses defined on the entity record as the shipping or billing address for the transaction. For this reason, setting a value for a shipping or billing address is in some cases slightly different from the way that you set other address summary fields. For details, see [Addresses Can Be Sourced from Entity Records](#).

The Address Subrecord Can Have Custom Forms

Compared with other subrecords, the address subrecord is unique in that you can create custom entry forms for it. For example, you may want to create custom forms for different countries.

If your account has multiple address forms, and if you are not seeing the expected results, the reason could be related to the form. In particular, if your script is using dynamic mode, the first value you set on the subrecord form should be country. The reason is that if you set a value for country that differs from the default, the form resets when the country value changes. Therefore, as a best practice, set the country value first.

Address Data Is Summarized on the Parent Record

With most types of subrecords, details that you enter on the subrecord are not summarized on the parent record when you view it in the UI. You have to open the subrecord in its own window to view its data.

The address subrecord is an exception to this rule. After you enter details into an address subrecord and return to the main view of the record, typically the system displays a summary of the details you entered. For example, in the following screenshot, subrecord data is summarized in the Address column.

Relationships	Communication	Address	Sales	Marketing	Support	Financial	Preferences	System Information	≡
ID	DEFAULT SHIPPING	DEFAULT BILLING	RESIDENTIAL ADDRESS			LABEL	ADDRESS	EDIT	
2359	Yes	Yes				Company Headquarters	2955 Campus Drive Suite 100 San Mateo CA 94403-2511 United States		

This summary represents the value of one field on the address subrecord called `addrtext`. This value is created through the use of a template and generated from other values entered on the subrecord, such as the values for the city and state fields. Each address form can have its own template for determining the `addrtext` value. You can view the template for any address form by viewing its record at [Customization > Forms > Address Forms](#).

An error in the `addrtext` field does not necessarily signify an error in the other values saved to the subrecord. If an error exists in the summary, review the template to make sure that it is capturing the values you intend.

To view the values for all of the fields on the subrecord, do one of the following:

- Use one of the following methods to retrieve the subrecord: `getCurrentSublistSubrecord()`, `getSublistSubrecord()`, or `getSubrecord()`. For an example, see [Example: Retrieving an Address Subrecord](#).
- In the UI, open the subrecord for editing in its own window.

Subrecord Scripting in SuiteScript 2.0 Compared With 1.0

Compared with SuiteScript 1.0, SuiteScript 2.0 introduces the following changes in how you script subrecords:

- A Single Method Lets You Create and Load Subrecords
- Subrecords Do Not Have to Be Explicitly Saved
- To Create Addresses, You Must Use Subrecord Methods

A Single Method Lets You Create and Load Subrecords

In SuiteScript 1.0, you use one set of APIs to create subrecords and another set to edit subrecords. For example, you could use nlapiCreateSubrecord to create a subrecord. You could use nlapiEditSubrecord to edit a subrecord.

By contrast, in SuiteScript 2.0, any method that creates a subrecord can also be used to load that subrecord for the purpose of editing it. These methods all have the word *get* in their names. For example, you use the *getSubrecord()* method to create or load a subrecord that exists on the body of a record. You use the *getSublistSubrecord()* or *getCurrentSublistSubrecord()* method to create or load a subrecord that exists on a sublist line.

When you use any of these methods, the system responds with the following logic:

- If a subrecord instance already exists in the specified field, the subrecord is loaded.
- If no subrecord instance exists, the system creates one. You can then set values on the field. The subrecord is saved when you save the record.

Subrecords Do Not Have to Be Explicitly Saved

In SuiteScript 1.0, you had to explicitly save a subrecord prior to saving the record. However, in SuiteScript 2.0, after you create a subrecord or make changes to one, you are not required to explicitly save the subrecord (and no methods exist for that purpose). Your new subrecord is saved at the time you save the record. The same rule applies if you make changes to an existing subrecord. Your updates are saved at the time you save the record.

To Create Addresses, You Must Use Subrecord Methods

The address subrecord was introduced in version 2014.2. Prior to that time, each address was represented on a record as a series of body fields or as a line in a sublist.

After the introduction of the address subrecord, SuiteScript 1.0 was enhanced to support two methods of interacting with addresses: In 1.0, you have the choice of interacting with addresses using subrecord APIs, which is the preferred method. But in 1.0 you can also interact with addresses using the legacy approach: setting values for the address body and sublist fields that used to exist. This support was made possible by logic added to the system that read the values set in this manner and created an address subrecord on behalf of the 1.0 script. Because this support exists in 1.0, these deprecated fields are displayed in the SuiteScript Records Browser as available fields.

However, in SuiteScript 2.0, to create an address, you **must** use subrecord methods. The system does not provide logic for the legacy address body and sublist fields. For that reason, to create, edit, or load an address in SuiteScript 2.0, you must instantiate the address subrecord by referencing the appropriate summary field.

Scripting Subrecords that Occur on Sublist Lines

In many cases, a subrecord is accessed through a field on a sublist line. For example:

- Every purchase order must include a list of items. Depending on the configuration of each item, the sublist line may be required to include an inventory detail subrecord.

- A vendor bill may include a list of items. If the bill is configured to track landed cost per line, each line in the Items sublist may include a landed cost subrecord.
- An employee may have multiple addresses. Each address is stored in a subrecord, and each subrecord is associated with a line in the Address sublist.

In each case, the sublist line also has fields that are not part of the subrecord. The subrecord itself is associated with only one field on the sublist line. In the SuiteScript Records Browser, this field is always identified as a field of type summary.

For more details, see the following sections:

- [Using SuiteScript 2.0 to Create a Subrecord in a Sublist Field](#)
- [Using SuiteScript 2.0 to Edit a Subrecord that Occurs in a Sublist Field](#)
- [Using SuiteScript 2.0 to Retrieve a Sublist Subrecord](#)

Using SuiteScript 2.0 to Create a Subrecord in a Sublist Field

Depending on the record type and other variables, a line on a record's sublist can include a field that references a subrecord. In many cases, you must add the subrecord at the time you are creating the sublist line. In other cases, you can go back and add the subrecord later.

To create a sublist subrecord, your script must use the Record module. The script can use either dynamic or standard mode. For details, see the following sections:

- [Creating a Sublist Subrecord in Dynamic Mode](#)
- [Creating a Sublist Subrecord in Standard Mode](#)

Subrecords can also occur in the body field of a record. For details on working with subrecords when occur in body fields, see [Scripting Subrecords that Occur in Body Fields](#). For an overview of the difference between these two types of placement, see [Body Field Subrecords and Sublist Subrecords](#).



Note: For more details about the methods referenced in this topic, see [Record Object Members](#).

Creating a Sublist Subrecord in Dynamic Mode

If your script uses dynamic mode, you can use the following procedure to create a subrecord in a sublist field.

To create a sublist subrecord in dynamic mode:

1. If you are adding a new record, create it and set the required body fields. If you are updating an existing record, load the record.
2. Do one of the following:
 - If you are creating a new sublist line, create the line using the `selectNewLine()` method. Set any required values on the sublist line by using the `setCurrentSublistValue()` method.
 - If you are adding a new subrecord to an existing sublist line, identify that line using the `selectLine()` method.
3. Create the new subrecord by using the `getCurrentSublistSubrecord()` method. This method takes two arguments:
 - A sublistId, which identifies the sublist.
 - A fieldId, which identifies the field on the sublist that contains the subrecord. In the Records Browser, the field that holds the subrecord is always identified as a field of type summary.

For example, you could use an expression like the following to create an order schedule subrecord on an item sublist:

```
...
var orderScheduleSubrecord = blanketPurchaseOrder.getCurrentSublistSubrecord({
    sublistId: 'item',
    fieldId: 'orderschedule'
});
...
```

4. As appropriate, set body fields on the subrecord by using the `setValue()` method. For example, on an order schedule subrecord, you could use the following expression to set a value for the Create Purchase Orders select field.

```
...
orderScheduleSubrecord.setValue({
    fieldId: 'createpurchaseorder',
    value: 'LEAD'
});
...
```

However, note that some subrecords do not have writable body fields.

5. If the subrecord has a sublist, generally you are required to add at least one line to the sublist. For each line, use the following guidelines:
 - Create the line by using the `selectNewLine()` method.
 - Set required values on the line by using the `setCurrentSublistValue()` method.
 - Save the subrecord's sublist line by using the `commitLine()` method.

For example, if you were creating an order schedule subrecord, you could use the following expressions to create a line on the subrecord's schedule sublist:

```
...
orderScheduleSubrecord.selectNewLine
    sublistId: 'schedule',
);
orderScheduleSubrecord.setCurrentSublistValue({
    sublistId: 'schedule',
    fieldId: 'quantity',
    value: 1
});

orderScheduleSubrecord.setCurrentSublistValue({
    sublistId: 'schedule',
    fieldId: 'trandate',
    value: dateVariable
});

orderScheduleSubrecord.commitLine({
    sublistId: 'schedule'
});
...
```

6. Save the sublist line that holds the subrecord by using the `commitLine()` method.

7. Save the record by using the save() method.

Note: For a full script sample, see [Example: Creating an Inventory Detail Sublist Subrecord](#) and [Example: Creating an Address Sublist Subrecord](#).

Creating a Sublist Subrecord in Standard Mode

If your script uses standard mode, you can use the following procedure to create a sublist field.

To create a sublist subrecord in standard mode:

1. If you are adding a new record, create it and set the required body fields. If you are updating an existing record, load the record.
2. If you are creating a new sublist line, create the line by using the insertLine() method. Set any required fields on the sublist line.
3. Create the new sublist by using the getSublistSubrecord() method. This method takes three arguments:
 - A sublistId, which identifies the sublist.
 - A fieldId, which identifies the field on the sublist that contains the subrecord. In the Records Browser, the field that holds the subrecord is always identified as a field of type summary.
 - A line number.

For example, you could use an expression like the following to create an inventory detail subrecord on the first line in an item sublist:

```
...
inventoryDetailSubrecord = rec.getSublistSubrecord({
    sublistId: 'item',
    fieldId: 'inventorydetail',
    line: 0
});
```

4. As appropriate, set body fields on the subrecord by using the setValue() method. However, note that some subrecords do not have writable body fields.
5. If the subrecord has a sublist, generally you are required to add at least one line to the sublist. For each line, use the following guidelines:
 - Use the insertLine() method to create the line.
 - Set values on the line by using the setSublistValue method.

For example, if you were creating an order schedule subrecord, you could use the following expressions to create a line on the subrecord's schedule sublist:

```
...
subrecordInvDetail.setSublistValue({
    sublistId: 'inventoryassignment',
    fieldId: 'receiptinventorynumber',
    line: 0,
    value: '012345'
});
```

6. Save the record by using the save() method.



Note: For a full script sample of creating a sublist subrecord in standard mode, see [Example: Creating a Landed Cost Sublist Subrecord](#).

Example: Creating an Inventory Detail Sublist Subrecord

The following example shows how to create a purchase order record that includes an inventory detail sublist subrecord. The script adds one line to the item sublist and creates an inventory detail subrecord on that line.

To use this sample, you must meet the following prerequisites:

- The Advanced Bin / Numbered Inventory Management feature must be enabled at Setup > Company > Enable Features, on the Items & Inventory subtab.
- The item you add to the sublist should be a lot-numbered inventory item.
- The receiptinventorynumber value must be unique in your system.

This example uses dynamic mode, but you could also add the subrecord using standard mode. For general details about using either approach to add a sublist subrecord, see [Using SuiteScript 2.0 to Create a Subrecord in a Sublist Field](#).

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

define([ 'N/record' ],

    function(record) {

        function afterSubmit(context) {

            // Create the purchase order.

            var rec = record.create({
                type: record.Type.PURCHASE_ORDER,
                isDynamic: true
            });

            // Set body fields on the purchase order.

            rec.setValue({
                fieldId: 'entity',
                value: '1663'
            });

            rec.setValue({
                fieldId: 'location',
                value: '6'
            });

            // Create one line in the item sublist.

        }

    }

);
```

```

rec.selectNewLine({
    sublistId: 'item'
});

rec.setCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    value: '299'
});

rec.setCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'quantity',
    value: 1
});

// Create the subrecord for that line.

var subrec = rec.getCurrentSublistSubrecord({
    sublistId: 'item',
    fieldId: 'inventorydetail'
});

// Add a line to the subrecord's inventory assignment sublist.

subrec.selectNewLine({
    sublistId: 'inventoryassignment'
});

subrec.setCurrentSublistValue({
    sublistId: 'inventoryassignment',
    fieldId: 'quantity',
    value: 2
});

subrec.setCurrentSublistValue({
    sublistId: 'inventoryassignment',
    fieldId: 'receiptinventorynumber',
    value: '01234'
});

// Save the line in the subrecord's sublist.

subrec.commitLine({
    sublistId: 'inventoryassignment'
});

// Save the line in the record's sublist

rec.commitLine({
    sublistId: 'item'
}

```

```

    });

    // Save the record.

    try {

        var recId = rec.save();

        log.debug({
            title: 'Record created successfully',
            details: 'Id: ' + recId
        });

    } catch (e) {

        log.error({
            title: e.name,
            details: e.message
        });
    }
}

return {
    afterSubmit: afterSubmit
};
});

```

Example: Creating an Address Sublist Subrecord

The following example shows how to create an employee record and populate the Address sublist with one line. The script also creates an address subrecord on the sublist line.

This example uses dynamic mode, but you could also add the subrecord using standard mode. For general details about using either approach to add a sublist subrecord, see [Using SuiteScript 2.0 to Create a Subrecord in a Sublist Field](#).

```

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

define([ 'N/record' ],

    function(record) {

        function afterSubmit(context) {

            // Create the record.

            var rec = record.create({
                type: record.Type.EMPLOYEE,
                isDynamic: true
            });

```

```
// Set the required body fields.

rec.setValue({
    fieldId: 'firstname',
    value: 'John'
});

rec.setValue({
    fieldId: 'lastname',
    value: 'Smith'
});

rec.setValue({
    fieldId: 'subsidiary',
    value: '1'
});

// Create a line in the Address sublist.

rec.selectNewLine({
    sublistId: 'addressbook'
});

// Set an optional field on the sublist line.

rec.setCurrentSublistValue({
    sublistId: 'addressbook',
    fieldId: 'label',
    value: 'Primary Address'
});

// Create an address subrecord for the line.

var subrec = rec.getCurrentSublistSubrecord({
    sublistId: 'addressbook',
    fieldId: 'addressbookaddress'
});

// Set body fields on the subrecord. Because the script uses
// dynamic mode, you should set the country value first. The country
// value determines which address form is to be used, so by setting
// this value first, you ensure that the values for the rest
// of the form's fields will be set properly.

subrec.setValue({
    fieldId: 'country',
    value: 'US'
});
```

```

        subrec.setValue({
            fieldId: 'city',
            value: 'San Mateo'
        });

        subrec.setValue({
            fieldId: 'state',
            value: 'CA'
        });

        subrec.setValue({
            fieldId: 'zip',
            value: '94403'
        });

        subrec.setValue({
            fieldId: 'addr1',
            value: '2955 Campus Drive'
        });

        subrec.setValue({
            fieldId: 'addr2',
            value: 'Suite 100'
        });

        // Save the sublist line.

        rec.commitLine({
            sublistId: 'addressbook'
        });

        // Save the record.

        try {
            var recId = rec.save();

            log.debug({
                title: 'Record created successfully',
                details: 'Id: ' + recId
            });
        } catch (e) {

            log.error({
                title: e.name,
                details: e.message
            });
        }
    }

    return {
        afterSubmit: afterSubmit
    };
}

```

```
});
```

Example: Creating an Order Schedule Sublist Subrecord

The following example creates a blanket purchase order record. It creates one line in the item sublist and creates an order schedule subrecord on that line.

To use this example, the Blanket Purchase Order feature must be enabled at Setup > Company > Enable Features, on the Transactions subtab.

This example uses dynamic mode, but you could also add the subrecord using standard mode. For general details about using either approach to add a sublist subrecord, see [Using SuiteScript 2.0 to Create a Subrecord in a Sublist Field](#).

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

define([ 'N/record' ],

    function(record) {

        function afterSubmit(context) {

            // Create the record.

            var rec = record.create({
                type: record.Type.BLANKET_PURCHASE_ORDER,
                isDynamic: true
            });

            // Set body fields on the record.

            rec.setValue({
                fieldId: 'entity',
                value: '1663'
            });

            rec.setValue({
                fieldId: 'location',
                value: '6'
            });

            rec.setValue({
                fieldId: 'memo',
                value: '456789'
            });

            // Create one line in the item sublist.

            rec.selectNewLine({
                sublistId: 'item',

```

```

        line: 0
    });

    rec.setCurrentSublistValue({
        sublistId: 'item',
        fieldId: 'item',
        value: '500'
    });

    rec.setCurrentSublistValue({
        sublistId: 'item',
        fieldId: 'quantity',
        value: '1'
    });

// Create the subrecord for that line.

var subrec = rec.getCurrentSublistSubrecord({
    sublistId: 'item',
    fieldId: 'orderschedule'
});

// Set a field on the body of the subrecord.

subrec.setValue({
    fieldId: 'createpurchaseorder',
    value: 'LEAD'
});

// Create a line in the subrecord's sublist.

subrec.selectNewLine({
    sublistId: 'schedule',
});

subrec.setCurrentSublistValue({
    sublistId: 'schedule',
    fieldId: 'quantity',
    value: 1
});

var nextQuarter = new Date();
nextQuarter.setDate(nextQuarter.getDate() + 90);

subrec.setCurrentSublistValue({
    sublistId: 'schedule',
    fieldId: 'trandate',
    value: nextQuarter
});

// Save the line in the subrecord's sublist.

```

```

    subrec.commitLine({
        sublistId: 'schedule'
    });

    // Save the line in the record's sublist

    rec.commitLine({
        sublistId: 'item'
    });

    // Save the record.

    try {
        var recId = rec.save();

        log.debug({
            title: 'Record created successfully',
            details: 'Id: ' + recId
        });
    } catch (e) {

        log.error({
            title: e.name,
            details: e.message
        });
    }
}

return {
    afterSubmit: afterSubmit
};
);

```

Example: Creating a Landed Cost Sublist Subrecord

The following example creates a vendor bill record. This example creates one line in the item sublist. It also sets the Landed Cost per Line option to true. With this configuration, it is possible to create a landed cost subrecord for each line.

To use this sample, the Landed Cost feature must be enabled at Setup > Company > Enable Features, on the Items & Inventory subtab.

This example uses standard mode, but you could also add the subrecord using dynamic mode. For general details about using either approach to add a sublist subrecord, see [Using SuiteScript 2.0 to Create a Subrecord in a Sublist Field](#).

```

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

```

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

function(record) {

    function afterSubmit(context) {

        // Create the record.

        var rec = record.create({
            type : record.Type.VENDOR_BILL,
            isDynamic: false
        });

        // Set body fields on the record.

        rec.setValue({
            fieldId: 'entity',
            value: '1663'
        });

        rec.setValue({
            fieldId: 'location',
            value: '6'
        });

        rec.setValue({
            fieldId: 'tranid',
            value: '101A'
        });

        // Set the Landed Cost per Line field to true.

        rec.setValue({
            fieldId: 'landedcostperline',
            value: true
        });

        // Add an item to the Item sublist.

        rec.insertLine({
            sublistId: 'item',
            line: 0
        });

        // Set values on the sublist line.

        rec.setSublistValue({
            sublistId: 'item',
            fieldId: 'item',
            line: 0,
```

```

        value: '599'
    });

    rec.setSublistValue({
        sublistId: 'item',
        fieldId: 'quantity',
        line: 0,
        value: 1
    });

    rec.setSublistValue({
        sublistId: 'item',
        fieldId: 'location',
        line: 0,
        value: '6'
    });

}

// Create the subrecord.

var subrec = rec.getSublistSubrecord({
    sublistId: 'item',
    fieldId: 'landedcost',
    line: 0
});

// Add a line to the subrecord's Landed Cost Data sublist.

subrec.insertLine({
    sublistId: 'landedcostdata',
    line: 0
});

// Set values on the subrecord's sublist line.

subrec.setSublistValue({
    sublistId: 'landedcostdata',
    fieldId: 'costcategory',
    line: 0,
    value: 2
});

subrec.setSublistValue({
    sublistId: 'landedcostdata',
    fieldId: 'amount',
    line: 0,
    value: 17.85
});

// Save the record.

try {

```

```

        var recId = rec.save();

        log.debug({
            title: 'Record created successfully',
            details: 'Id: ' + recId
        });
    } catch (e) {
        log.error({
            title: e.name,
            details: e.message
        });
    }
}

return {
    afterSubmit: afterSubmit
};
);

```

Using SuiteScript 2.0 to Edit a Subrecord that Occurs in a Sublist Field

In some cases, your script can make changes to a subrecord that occurs in a sublist field.

To edit a sublist subrecord, your script must use the Record module. The script can use either dynamic or standard mode. For details, see the following sections:

- Editing a Subrecord in Dynamic Mode
- Editing a Subrecord in Standard Mode



Note: For more details about the methods referenced in this topic, see [Record Object Members](#).

Editing a Subrecord in Dynamic Mode

If your script uses dynamic mode, you can use the following procedure to edit a subrecord that occurs in a sublist field.

To edit a subrecord in dynamic mode:

1. Load the record.
2. Use the selectLine() method to identify the sublist and line that contain the subrecord that you want to update.
3. Retrieve the subrecord by using the getCurrentSublistSubrecord() method. This method takes two arguments:
 - A sublistId.
 - A fieldId, which identifies the field on the sublist that contains the subrecord. In the Records Browser, the field that holds the subrecord is always identified as a field of type summary.

For example, suppose you are working with an entity record, such as an employee or customer. You could use an expression like the following to load an address subrecord from the entity's Address sublist:

...

```

var addressSubrecord = rec.getCurrentSublistSubrecord({
    sublistId: 'addressbook',
    fieldId: 'addressbookaddress'
});
...

```

4. As appropriate, update body fields on the subrecord by using the setValue() method. For example, you could use an expression like the following to update a value on the address subrecord:

```

...
addressSubrecord.setValue({
    fieldId: 'city',
    value: 'St. Petersburg'
});
...

```

However, note that some subrecords do not have writable body fields.

5. If the subrecord has a sublist whose values you want to modify, use the following steps for each line you want to change:
 1. Identify the line you want to change by using the selectLine() method.
 2. For each value you want to change, use the setCurrentSublistValue() method to identify the field and the new value.
 3. Save your changes to the subrecord's sublist line by using the commitLine() method.
6. Save the line that holds the subrecord by using the commitLine() method.
7. Save the record by using the save() method.

Note: For a full script sample that shows editing a sublist subrecord in dynamic mode, see [Example: Updating an Order Schedule Sublist Subrecord](#).

Editing a Subrecord in Standard Mode

If your script uses standard mode, you can use the following procedure to edit a subrecord that occurs in a sublist field.

To edit a subrecord in standard mode:

1. Load the record.
2. Retrieve the subrecord by using the getSublistSubrecord() method. This method takes three arguments:
 - A sublistId.
 - A fieldId, which identifies the field on the sublist that contains the subrecord. In the Records Browser, the field that holds the subrecord is always identified as a field of type summary.
 - A line number, which identifies the sublist line that contains the subrecord you want to change.

For example, you could use an expression like the following to load an inventory detail subrecord from an item sublist:

```

...
inventoryDetailSubrecord = rec.getSublistSubrecord({
    sublistId: 'item',
    fieldId: 'inventorydetail',
}

```

```
    line: 0
});
...

```

3. As appropriate, update body fields on the subrecord by using the `setValue()` method. However, note that some subrecords do not have writable body fields.
4. If the subrecord has a sublist whose values you want to modify, use the `setSublistValue()` method to update the appropriate value. This method takes four arguments:
 - A sublistId.
 - A fieldId, which identifies the field you want to change.
 - A line number, which identifies the sublist line you want to change.
 - The new value.

For example, if you were updating an inventory detail subrecord, you could use the following expression to update the serial number on the first line of the inventory assignment sublist:

```
...
inventoryDetailSubrecord.setSublistValue({
    sublistId: 'inventoryassignment',
    fieldId: 'receiptinventorynumber',
    line: 0,
    value: '56789'
});
...

```

5. Save the record by using the `save()` method.

Note: For a full script sample showing how to modify a subrecord in standard mode, see [Example: Updating an Address Subrecord](#).

Example: Updating an Order Schedule Sublist Subrecord

The following example loads an existing blanket purchase order record. It selects a line on the record's item sublist, retrieves the subrecord associated with that line, and makes changes to the subrecord.

To use this sample, you must meet the following prerequisites:

- The Blanket Purchase Order feature must be enabled at Setup > Company > Enable Features, on the Transactions subtab.
- The blanket purchase order record that you reference should have at least one line in the item sublist. The line should reference an existing order schedule subrecord.

This example uses dynamic mode, but you could also update the subrecord using standard mode. For general details about using either approach to update a sublist subrecord, see [Using SuiteScript 2.0 to Edit a Subrecord that Occurs in a Sublist Field](#).

```
define([ 'N/record' ],
function(record) {
    function afterSubmit(context) {
        // Load the blanket purchase order record.
    }
});
```

```
var rec = record.load({
    type: record.Type.BLANKET_PURCHASE_ORDER,
    id: 3319,
    isDynamic: true
});

// Select the sublist and line.

rec.selectLine({
    sublistId: 'item',
    line: 0
});

// Retrieve the subrecord.

var subrec = rec.getCurrentSublistSubrecord({
    sublistId: 'item',
    fieldId: 'orderschedule'
});

// Select the appropriate line in the subrecord's sublist.

subrec.selectLine({
    sublistId: 'schedule',
    line: 0
});

// Identify the field to be modified, and set new value.

var nextQuarter = new Date();
nextQuarter.setDate(nextQuarter.getDate() + 90);

subrec.setCurrentSublistValue({
    sublistId: 'schedule',
    fieldId: 'trandate',
    value: nextQuarter
});

// Save the subrecord's sublist line.

subrec.commitLine({
    sublistId: 'schedule'
});

// Save the item sublist line that contains the subrecord.

rec.commitLine({
    sublistId: 'item'
```

```

    });

    // Save the record.

    try {
        var recId = rec.save();

        log.debug({
            title: 'Record updated successfully',
            details: 'Id: ' + recId
        });
    } catch (e) {

        log.error({
            title: e.name,
            details: e.message
        });
    }
}

return {
    afterSubmit: afterSubmit
};
);
}

```

Example: Updating an Address Subrecord

The following example shows how to update an employee address. On the employee record, each address is stored in a subrecord that is associated with a line in the Address sublist.

This example uses standard mode, but you could also edit the subrecord using dynamic mode. For general details about using either approach to update a sublist subrecord, see [Using SuiteScript 2.0 to Edit a Subrecord that Occurs in a Sublist Field](#).

```

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

define([ 'N/record' ],

    function(record) {

        function afterSubmit(context) {

            // Load the record.

            var rec = record.load({
                type: record.Type.EMPLOYEE,
                id: 1863,
                isDynamic: false
            });

```

```

// Retrieve the sublist to be modified.

var subrec = rec.getSublistSubrecord({
    sublistId: 'addressbook',
    fieldId: 'addressbookaddress',
    line: 0
});

// Change a field on the sublist.

subrec.setValue({
    fieldId: 'addr1',
    value: '15 Main Street'
});

// Save the record.

try {
    var recId = rec.save();

    log.debug({
        title: 'Record updated successfully',
        details: 'Id: ' + recId
    });
}

} catch (e) {

    log.error({
        title: e.name,
        details: e.message
    });
}

return {
    afterSubmit: afterSubmit
};
});

```

Using SuiteScript 2.0 to Retrieve a Sublist Subrecord

In some cases, you may want to retrieve data from a sublist subrecord that occurs in a sublist field.

To retrieve a sublist subrecord, your script can use the Record module. Your script can use either dynamic or standard mode. For details, see the following sections:

- [Retrieving a Sublist Subrecord in Dynamic Mode](#)
- [Retrieving a Subrecord in Standard Mode](#)



Note: For more details about the methods referenced in this topic, see [Record Object Members](#).

Retrieving a Sublist Subrecord in Dynamic Mode

If your script uses dynamic mode, you can use the following procedure to retrieve a sublist subrecord that occurs in a sublist field.

To retrieve a sublist subrecord in dynamic mode:

1. Load the parent record.
2. Select the line that contains the subrecord by using the `selectLine()` method.
3. Retrieve the subrecord by using the `getCurrentSublistSubrecord()` method. This method takes two arguments:
 - A sublistId.
 - A fieldId, which identifies the field on the sublist that contains the subrecord. In the Records Browser, the field that holds the subrecord is always identified as a field of type summary.

For example, suppose you are working with an entity record, such as an employee or customer. You could use an expression like the following to load an address subrecord from the entity's Address sublist:

```
...
var addressSubrecord = rec.getCurrentSublistSubrecord({
  sublistId: 'addressbook',
  fieldId: 'addressbookaddress',
  line: 0
});
```

4. If you want to retrieve a value from the body of the subrecord, use the `getValue()` method. For example, you could use an expression like the following to retrieve a detail from the body of an address subrecord:

```
...
var cityValue = addressSubrecord.getValue({
  fieldId: 'city'
});
```

5. If you want to retrieve a value from the subrecord's sublist, use the `getSublistValue()` method.



Note: For a full script sample showing how to retrieve a sublist subrecord in dynamic mode, see [Example: Retrieving an Order Schedule Subrecord](#).

Retrieving a Subrecord in Standard Mode

If your script uses standard mode, use the following procedure to retrieve a sublist subrecord.

To retrieve a subrecord in standard mode

1. Load the parent record.
2. Retrieve the subrecord by using the `getSublistSubrecord()` method. This method takes three arguments:

- A sublistId.
- A fieldId, which identifies the field on the sublist that contains the subrecord. In the Records Browser, the field that holds the subrecord is always identified as a field of type **summary**.
- A line number,

For example, you could use an expression like the following to load an order schedule subrecord from a blanket purchase order record:

```
...
var orderScheduleSubrecord = rec.getSublistSubrecord({
    sublistId: 'item',
    fieldId: 'orderschedule',
    line: 0
});
...
...
```

3. If you want to retrieve a value from the body of the subrecord, use the `getValue()` method.
4. If you want to retrieve a value from the subrecord's sublist, use the `getSublistValue()` method.

For example, the order schedule subrecord has a schedule sublist. If you wanted to retrieve the date value from the second line of the schedule sublist, you would use an expression like the following:

```
...
var dateValue = orderScheduleSubrecord.getSublistValue({
    sublistId: 'schedule',
    fieldId: 'trandate',
    line: 1
});
...
...
```



Note: For a full script sample showing how to retrieve a sublist subrecord in standard mode, see [Example: Retrieving an Address Subrecord](#).

Example: Retrieving an Order Schedule Subrecord

The following example loads a blanket purchase order record. It selects a line on the item sublist, then retrieves the order schedule associated with that line. It reads two values from the subrecord and prints them to the script deployment execution log.

If you want to use this script, you must meet the following prerequisites:

- The Blanket Purchase Order feature must be enabled at Setup > Company > Enable Features, on the Transactions subtab.
- The blanket purchase order record you reference must have at least one line in the Item sublist, and that line should include an order schedule subrecord.

This example uses dynamic mode, but you could also load the subrecord using standard mode. For general details about using either approach to retrieve a sublist subrecord, see [Using SuiteScript 2.0 to Retrieve a Sublist Subrecord](#).

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

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

function(record) {

    function afterSubmit(context) {

        // Load the record.

        var rec = record.load({
            type: record.Type.BLANKET_PURCHASE_ORDER,
            id: 3120,
            isDynamic: true
        });

        // Retrieve the subrecord.

        rec.selectLine({
            sublistId: 'item',
            line: 0
        });

        var subrec = rec.getCurrentSublistSubrecord({
            sublistId: 'item',
            fieldId: 'orderschedule',
            line: 0
        });

        // Create a variable and initialize it to the
        // value of the trandate field.

        var dateValue = subrec.getSublistValue({
            sublistId: 'schedule',
            fieldId: 'trandate',
            line: 0
        });

        // Create a variable and initialize it to the
        // value of the memo field.

        var memoValue = subrec.getSublistValue({
            sublistId: 'schedule',
            fieldId: 'memo',
            line: 0
        });

        // Print the retrieved values to the execution log.

        try {

            log.debug({

```

```

        title: 'date value',
        details: 'date value: ' + dateValue
    });

    log.debug({
        title: 'memo value',
        details: 'memo value: ' + memoValue
    });

} catch (e) {

    log.error({
        title: e.name,
        details: e.message
    });
}

return {
    afterSubmit: afterSubmit
};
);
}

```

Example: Retrieving an Address Subrecord

This example loads an employee record and selects a line on the address sublist. It retrieves a value from the sublist line that is not part of the subrecord. It also retrieves the address subrecord and reads one of the subrecord's fields. The script prints both values to the script deployment record's execution log.

This example uses standard mode, but you could also load the subrecord using dynamic mode. For general details about using either approach to retrieve a sublist subrecord, see [Using SuiteScript 2.0 to Retrieve a Sublist Subrecord](#).

```

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

define([ 'N/record' ],

function(record) {

    function afterSubmit(context) {

        // Load the record.

        var rec = record.load({
            type: record.Type.EMPLOYEE,
            id: 1863,
            isDynamic: false
        });

        // Retrive a value from the sublist line
    }
});

```

```

var labelValue = rec.getSublistValue({
    sublistId: 'addressbook',
    fieldId: 'label',
    line: 0
});

// Retrieve the subrecord associated with that same line.

var subrec = rec.getSublistSubrecord({
    sublistId: 'addressbook',
    fieldId: 'addressbookaddress',
    line: 0
});

// Create a variable to initialize it to the
// value of the subrecord's city field.

var cityValue = subrec.getValue({
    fieldId: 'city'
});

// Print the retrieved values to the execution log.

try {

    log.debug({
        title: 'label value',
        details: 'label value: ' + labelValue
    });

    log.debug({
        title: 'city value',
        details: 'city value: ' + cityValue
    });

} catch (e) {

    log.error({
        title: e.name,
        details: e.message
    });
}

return {
    afterSubmit: afterSubmit
};
});

```

Scripting Subrecords that Occur in Body Fields

In some cases, a subrecord is accessed through a body field on a record.

Most subrecords referenced from body fields behave in the same general way. However, there are some differences when working with subrecords that are used as the shipping or billing addresses on transactions.

For details, see the following sections:

- [Body Field Subrecords](#)
- [Transaction Shipping and Billing Addresses](#)

Body Field Subrecords

The following are examples of typical subrecords that are referenced from body fields:

- An assembly build record may need to include the serial numbers of each assembly being tracked. These details would be stored in an inventory detail subrecord referenced from a body field on the assembly build record.
- A subsidiary record may include a main address, a shipping address, and a return address. Each of these addresses is referenced by a field on the body of the record.

For details about working with these types of subrecords, see the following sections:

- [Using SuiteScript 2.0 to Create a Body Field Subrecord](#)
- [Using SuiteScript to Update a Body Field Subrecord](#)
- [Using SuiteScript 2.0 to Retrieve a Body Field Subrecord](#)

Transaction Shipping and Billing Addresses

Some transactions can have body fields that represent shipping and billing addresses. Each of these fields contains an address subrecord. The process of interacting with these subrecords can be different from working with other body field subrecords. For details, see [Scripting Transaction Shipping and Billing Addresses](#).

Using SuiteScript 2.0 to Create a Body Field Subrecord

Some types of records have body fields that can reference subrecords. Scripting a subrecord that exists in a body field is slightly different from working with those that exist on sublists.

To create a subrecord, your script must use the Record module. The script can use either dynamic or standard mode. For details on each approach, see the following sections:

- [Creating a Body Field Subrecord in Dynamic Mode](#)
- [Creating a Body Field Subrecord in Standard Mode](#)



Note: For more details about the methods referenced in this topic, see [Record Object Members](#).



Important: If you are scripting the shipping address or billing address on a transaction, see [Scripting Transaction Shipping and Billing Addresses](#).

Creating a Body Field Subrecord in Dynamic Mode

If your script uses dynamic mode, you can use the following procedure to create a subrecord in a body field.

To create a body field subrecord in dynamic mode:

1. If you are adding a new record, create it and set the required body fields. If you are updating an existing record, load the record.
2. Create the subrecord by using the `getSubrecord()` method. This method takes one argument: A `fieldId`, which identifies the field on the record that contains the subrecord. In the Records Browser, the field that holds the subrecord is always identified as a field of type summary.

For example, you could use an expression like the following to create an inventory detail subrecord on an assembly build record:

```
...
var inventoryDetailSubrecord = rec.getSubrecord({
    fieldId: 'inventorydetail'
});
...
```

3. As appropriate, set body fields on the subrecord using the `setValue()` method. However, note that some subrecords do not have writable body fields.
4. If the subrecord has a sublist, generally you are required to add at least one line to the sublist. For each line, use the following guidelines:
 - Create the line by using the `selectNewLine()` method.
 - Set required values on the line by using the `setCurrentSublistValue()` method.
 - Save the subrecord's sublist line by using the `commitLine()` method.

For example, if you were creating an inventory detail subrecord, you could use the following expression to create a line on the subrecord's inventory assignment sublist:

```
...
inventoryDetailSubrecord.selectNewLine({
    sublistId: 'inventoryassignment',
});

inventoryDetailSubrecord.setCurrentSublistValue({
    sublistId: 'inventoryassignment',
    fieldId: 'receiptinventorynumber',
    value: '012345'
});

inventoryDetailSubrecord.commitLine({
    sublistId: 'inventoryassignment'
})
...
```

5. Save the record.



Note: For full script samples showing how to create a body field subrecord using dynamic mode, see [Example: Creating an Address on a Subsidiary Record](#) and [Example: Creating an Inventory Detail Subrecord on a Body Field](#).

Creating a Body Field Subrecord in Standard Mode

If your script uses standard mode, use the following procedure to create a subrecord on the body field of a record.

To create a body field subrecord in standard mode:

1. If you are adding a new record, create it and set the required body fields. If you are updating an existing record, load the record.
2. Create the subrecord by using the `getSubrecord()` method. This method takes one argument: A `fieldId`, which identifies the body field on the record that contains the subrecord. In the Records Browser, the field that holds the subrecord is always identified as a field of type `summary`.

For example, you could use an expression like the following to create an address subrecord on a location record:

```
...
var addressSubrecord = rec.getSubrecord({
    fieldId: 'mainaddress'
});
...
```

3. As appropriate, set body fields on the subrecord using the `setValue()` method. However, note that some subrecords do not have writable body fields.
4. If the subrecord has a sublist, generally you are required to add at least one line to the sublist. For each line, use the following guidelines:
 - Create the line by using the `insertLine()` method.
 - Set required values on the line by using the `setSublistValue()` method.

For example, if you were creating an inventory detail subrecord, you could use the following expressions to create a line on the subrecord's inventory assignment sublist:

```
...
inventoryDetailSubrecord.insertLine({
    sublistId: 'inventoryassignment',
    line: 0
});

inventoryDetailSubrecord.setSublistValue({
    sublistId: 'inventoryassignment',
    fieldId: 'receiptinventorynumber',
    line: 0,
    value: '12345'
});
...
```

5. Save the record.

Example: Creating an Address on a Subsidiary Record

The following example shows how to create a subsidiary record that includes an address. In this case, the address data is contained in an address subrecord assigned to the `mainaddress` field.

To use this sample, you must meet the following prerequisites:

- You must have a OneWorld account.
- The value you use for the subsidiary record's name field must be unique in your system.

This example uses dynamic mode, but you could also add the subrecord using standard mode. For general details about using either approach, see [Using SuiteScript 2.0 to Create a Body Field Subrecord](#).

```
/**
```

```

* @NApiVersion 2.x
* @NScriptType UserEventScript
*/
define([ 'N/record' ],
  function(record) {
    function afterSubmit(context) {
      var rec = record.create({
        type: record.Type.SUBSIDIARY,
        isDynamic: true
      });

      // Set body fields on the record.

      rec.setValue({
        fieldId: 'name',
        value: 'US Subsidiary'
      });

      rec.setValue({
        fieldId: 'state',
        value: 'CA'
      });

      // Create the address subrecord.

      var subrec = rec.getSubrecord({
        fieldId: 'mainaddress',
      });

      subrec.setValue({
        fieldId: 'city',
        value: 'San Mateo'
      });

      subrec.setValue({
        fieldId: 'state',
        value: 'CA'
      });

      subrec.setValue({
        fieldId: 'zip',
        value: '94403-2511'
      });

      subrec.setValue({
        fieldId: 'addr1',
        value: '2955 Campus Drive'
      });
    }
  }
);

```

```

        subrec.setValue({
            fieldId: 'addr2',
            value: 'Suite 100'
        });

        // Save the record.

        try {
            var recId = rec.save();

            log.debug({
                title: 'Record created successfully',
                details: 'Id: ' + recId
            });
        } catch (e) {

            log.error({
                title: e.name,
                details: e.message
            });
        }
    }

    return {
        afterSubmit : afterSubmit
    };
});

```

Example: Creating an Inventory Detail Subrecord on a Body Field

The following example shows how to create an assembly build record that includes an inventory detail subrecord. In this case, the subrecord is contained in a body field called inventory detail.

To use this sample, you must meet the following prerequisites:

- The Advanced Bin / Numbered Inventory Management feature must be enabled at Setup > Company > Enable Features, on the Items & Inventory subtab.
- The item selected for the item body field should be a serialized assembly item.
- The receiptinventorynumber value must be unique in your system.

This example uses dynamic mode, but you could also add the subrecord using standard mode. For general details about using either approach to add a sublist subrecord, see [Using SuiteScript 2.0 to Create a Body Field Subrecord](#).

```

/**
 * @NApiVersion 2.x
 * @NScriptType UserEventScript
 */
define([ 'N/record' ],

    function(record) {

```

```
function afterSubmit(context) {  
  
    // Create the record.  
  
    var rec = record.create({  
        type: record.Type.ASSEMBLY_BUILD,  
        isDynamic: true  
    });  
  
    // Set body fields.  
  
    rec.setValue({  
        fieldId: 'subsidiary',  
        value: '1'  
    });  
  
    rec.setValue({  
        fieldId: 'location',  
        value: '6'  
    });  
  
    rec.setValue({  
        fieldId: 'item',  
        value: '699'  
    });  
  
    rec.setValue({  
        fieldId: 'quantity',  
        value: 1  
    });  
  
    // Create the subrecord.  
  
    var subrec = rec.getSubrecord({  
        fieldId: 'inventorydetail'  
    });  
  
    // Create a line on the subrecord's inventory assignment sublist.  
  
    subrec.selectNewLine({  
        sublistId: 'inventoryassignment',  
    });  
  
    subrec.setCurrentSublistValue({  
        sublistId: 'inventoryassignment',  
        fieldId: 'receiptinventorynumber',  
        value: '012345'  
    });  
  
    subrec.commitLine({  
        sublistId: 'inventoryassignment'  
    });
```

```

// Save the record.

try {
    var recId = rec.save();

    log.debug({
        title: 'Record created successfully',
        details: 'Id: ' + recId
    });
}

} catch (e) {

    log.error({
        title: e.name,
        details: e.message
    });
}

return {
    afterSubmit: afterSubmit
};
});
}

```

Using SuiteScript 2.0 to Update a Body Field Subrecord

If the business logic of the parent record permits it, your script can load an existing body field subrecord and make changes to it.

To edit a subrecord, your script must use the Record module. The script can use either dynamic or standard mode.

To update a body field subrecord:

1. Load the record.
2. Retrieve the subrecord by using the `getSubrecord()` method. This method takes one argument: A `fieldId`, which identifies the field on the sublist that contains the subrecord. In the Records Browser, the field that holds the subrecord is always identified as a field of type summary.

For example, you could use an expression like the following to load the inventory detail subrecord associated with an assembly build record:

```

...
var subrec = call.getSubrecord({
    fieldId: 'inventorydetail'
});
...

```

3. As appropriate, update body fields on the subrecord by using the `setValue()` method. However, note that some subrecords do not have writable body fields.
4. If the subrecord has a sublist whose values you want to modify, do one of the following:
 - If your script uses dynamic mode, use the following steps for each value you want to change:
 1. Identify the line you want to change by using the `selectLine()` method.

2. For each value you want to change, use the `setCurrentSublistValue()` method to identify the field and the new value.
 3. Save your changes to the subrecord's sublist line by using the `commitLine()` method.
- If your script uses standard mode, use the `setSublistValue()` method to update each field that you want to change. This field takes four arguments:
 - A sublistId.
 - A fieldId, which identifies the field on the sublist that contains the subrecord.
 - A line number, which identifies the sublist line that contains the subrecord you want to change.
 - The new value.
5. Save the record by using the `save()` method.



Note: For full script samples showing how to edit a body field subrecord using dynamic mode, see [Example: Editing a Body Field Address Subrecord](#) and [Example: Editing a Body Field Inventory Detail Subrecord](#).

Example: Editing a Body Field Address Subrecord

The following example loads a subsidiary record. It also loads and edits the address subrecord stored in the subsidiary's mainaddress body field.

To use this script, you must have a OneWorld account.

For general details on updating body field subrecords, see [Using SuiteScript 2.0 to Update a Body Field Subrecord](#).

```
/** 
 * @NApiVersion 2.x
 * @NScriptType UserEventScript
 */
define([ 'N/record' ],

  function(record) {

    function afterSubmit(context) {

      // Load the record.

      var rec = record.load({
        type: record.Type.SUBSIDIARY,
        id: 1,
        isDynamic: true
      });

      // Load the subrecord.

      var subrec = rec.getSubrecord({
        fieldId: 'mainaddress'
      });

    }
  }
);
```

```

// Make changes to one field.

subrec.setValue({
    fieldId: 'addr1',
    value: '12331-A Riata Trace Parkway'
});

// Save the record.

try {
    var recId = rec.save();

    log.debug({
        title: 'Record updated successfully',
        details: 'Id: ' + recId
    });

} catch (e) {

    log.error({
        title: e.name,
        details: e.message
    });
}

return {
    afterSubmit : afterSubmit
};
});

```

Example: Editing a Body Field Inventory Detail Subrecord

The following example loads an assembly build record. It also loads the subrecord stored in the assembly build's Inventory Detail body field and saves a change to the subrecord.

To use this example, you must meet the following prerequisites:

- The Advanced Bin / Numbered Inventory Management feature must be enabled at Setup > Company > Enable Features, on the Items & Inventory subtab.
- The assembly build record that you load must already have an inventory detail subrecord.
- The new value you choose for receiptinventorynumber must be unique in your system.

For general details on updating body field subrecords, see [Using SuiteScript 2.0 to Update a Body Field Subrecord](#).

```

/**
 * @NApiVersion 2.x
 * @NScriptType UserEventScript
 */
define([ 'N/record' ],

function(record) {

    function afterSubmit(context) {

```

```
// Load the record.

var rec = record.load({
    type: record.Type.ASSEMBLY_BUILD,
    id: 4918,
    isDynamic: true
});

// Load the subrecord.

var subrec = rec.getSubrecord({
    fieldId: 'inventorydetail'
});

// Identify a line on the subrecord's sublist.

subrec.selectLine({
    sublistId: 'inventoryassignment',
    line: 0
});

// Make changes to one sublist field.

subrec.setCurrentSublistValue({
    sublistId: 'inventoryassignment',
    fieldId: 'receiptinventorynumber',
    value: '890123'
});

// Save the record.

try {
    var recId = rec.save();

    log.debug({
        title: 'Record updated successfully',
        details: 'Id: ' + recId
    });

    } catch (e) {

    log.error({
        title: e.name,
        details: e.message
    });
}

return {
```

```

        afterSubmit: afterSubmit
    );
});

```

Using SuiteScript 2.0 to Retrieve a Body Field Subrecord

In some cases, you may want to retrieve data from a subrecord that occurs in a body field.

To retrieve a subrecord, your script can use either the Record or the currentRecord module. Your script can use either dynamic or standard mode.

To use SuiteScript 2.0 to retrieve a body field subrecord:

1. Load the record.
2. Retrieve the subrecord by using the getSubrecord() method. This method takes one argument: A fieldId, which identifies the field on the sublist that contains the subrecord. In the Records Browser, the field that holds the subrecord is always identified as a field of type summary.

For example, you could use an expression like the following to load an address subrecord stored in the body field of a location record:

```

...
var subrec = call.getSubrecord({
    fieldId: 'mainaddress'
});
...

```

3. If you want to retrieve a value from the body of the subrecord, use the getValue() method. For example, you could use an expression like the following to retrieve a detail from the body of an address subrecord:

```

...
var cityValue = subrec.getValue({
    fieldId: 'city'
});
...

```

4. If you want to retrieve a value from the subrecord's sublist, use the getSublistValue() method. For example, you could use an expression like the following to retrieve a detail from the sublist of an inventory detail subrecord:

```

...
var cityValue = subrec.getValue({
    sublistId: 'inventoryassignment',
    fieldId: 'receiptinventorynumber',
    line: 4,
});

...

```

Scripting Transaction Shipping and Billing Addresses

A transaction's shipping address is stored in an address subrecord referenced from a body field on the transaction. A transaction's billing address is stored in the same way. The process of interacting with

these subrecords is similar to how you work with other subrecords referenced by body fields. However, some differences exist. For details, see the following sections:

- Addresses Can Be Sourced from Entity Records
- A Default Address May Be Used
- New Shipping and Billing Addresses Are Always Custom

 **Note:** See also [Understanding the Address Subrecord](#).

Addresses Can Be Sourced from Entity Records

Many transactions are associated with an entity. The entity record may already have addresses defined on its Address subtab. When working with a transaction, you can designate one of these existing addresses as the transaction's shipping or billing address.

If you choose to reference an existing address, your script does not have to create an address subrecord or use any subrecord methods. Instead, you can pick an address by using a select field that is associated with the summary field. For example, to designate an existing address as the shipping address on a sales order, you use the shipaddresslist select field. This field identifies the subrecord being used by the transaction's shippingaddress summary field. For an example of how to select an existing address using the shipaddresslist select field, see [Example: Using SuiteScript 2.0 to Select an Existing Shipping Address](#).

A Default Address May Be Used

An entity can have a default shipping or billing address. For these types of entities, if you create a transaction and you don't set a value for shipping or billing address, the entity's default is used.

If the entity has a default shipping or billing address that you want to completely override, use removeSubrecord() to clear the summary field before you set the values for your new subrecord. As an alternative, you can set the shipaddresslist field to null before setting your new address values. For an example, see [Example: Using SuiteScript 2.0 to Create a New Shipping Address](#).

New Shipping and Billing Addresses Are Always Custom

In the UI, if you are creating a sales order and you want to enter a new shipping address, you must select either New or Custom in the Ship to Select field. If you select New, the address you enter is also saved to the customer record. If you select Custom, the new address is not saved to the customer record. It is saved only on the transaction.

By contrast, in SuiteScript, you cannot choose between New or Custom. Any new shipping or billing address created by your script is treated as a Custom address. The address is saved on the transaction, but it cannot be saved to the entity record.

Example: Using SuiteScript 2.0 to Create a New Shipping Address

The following script creates a sales order record with one line in the item sublist. It also creates a shipping address for the transaction. It overrides any default shipping address that might be defined on the customer record.

 **Note:** For details about working with a transaction's shipping or billing address, see [Scripting Transaction Shipping and Billing Addresses](#).

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

```
define([ 'N/record' ],  
       function(record) {  
  
         function afterSubmit(context) {  
  
           // Create the record.  
  
           var rec = record.create({  
             type: record.Type.SALES_ORDER,  
             isDynamic: true  
           });  
  
           // Set body fields on the record.  
  
           rec.setValue({  
             fieldId: 'entity',  
             value: '2163'  
           });  
  
           rec.setValue({  
             fieldId: 'memo',  
             value: '102A'  
           });  
  
           // To prevent the system from overriding the shipping address defined  
           // in this script with a default shipping address (if one exists),  
           // set the shipaddresslist field to null.  
  
           rec.setValue({  
             fieldId: 'shipaddresslist',  
             value: null  
           });  
  
           // Create the subrecord.  
  
           var subrec = rec.getSubrecord({  
             fieldId: 'shippingaddress'  
           });  
  
           // Set values on the subrecord.  
  
           subrec.setValue({  
             fieldId: 'city',  
             value: 'New York'  
           });  
  
           subrec.setValue({  
             fieldId: 'state',  
             value: 'NY'  
           });  
         }  
       }  
     );  
   }  
 );
```

```

        value: 'New York'
    });

    subrec.setValue({
        fieldId: 'zip',
        value: '10018'
    });

    subrec.setValue({
        fieldId: 'addr1',
        value: '8 W 40th St.'
    });

    subrec.setValue({
        fieldId: 'country',
        value: 'US'
    });

// Create a line in the item sublist.

rec.selectNewLine({
    sublistId: 'item'
});

rec.setCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    value: '100'
});

rec.setCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'quantity',
    value: '11'
});

rec.commitLine({
    sublistId: 'item'
});

// Save the record.

try {
    var recId = rec.save();

    log.debug({
        title: 'Record created successfully',
        details: 'Id: ' + recId
    });

} catch (e) {

    log.error({

```

```

        title: e.name,
        details: e.message
    });
}

return {
    afterSubmit: afterSubmit
};
});
}

```

Example: Using SuiteScript 2.0 to Select an Existing Shipping Address

The following script creates a sales order record with one item in the item sublist. It also sets the value of the shipping address field to the value of an address from the customer record. Because the script selects an existing address subrecord, it does not use subrecord methods. It identifies the address subrecord by internal ID.

Note: For details about working with a transaction's shipping or billing address, see [Scripting Transaction Shipping and Billing Addresses](#).

```

/**
 * @NApiVersion 2.x
 * @NScriptType UserEventScript
 */
define([ 'N/record' ],

function(record) {

    function afterSubmit(context) {

        var rec = record.create({
            type: record.Type.SALES_ORDER,
            isDynamic: true

        });

        // Set body fields on the record.

        rec.setValue({
            fieldId: 'entity',
            value: '110'
        });

        // Set the shipping address to the value of
        // an address on the customer record.

        rec.setValue({
            fieldId: 'shipaddresslist',
            value: '760'
        });

    }
});

```

```

// Create one line in the item sublist.

rec.selectNewLine({
    sublistId: 'item'
});

rec.setCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    value: '100'
});

rec.setCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'quantity',
    value: '3'
});

rec.commitLine({
    sublistId: 'item'
});

// Save the record.

try {
    var recId = rec.save();

    log.debug({
        title: 'Record created successfully',
        details: 'Id: ' + recId
    });

    } catch (e) {

        log.error({
            title: e.name,
            details: e.message
        });
    }
}

return {
    afterSubmit: afterSubmit
};
});

```

Example: Retrieving Details from a Shipping or Billing Address

The following script loads a sales order record. It retrieves a value from the shipping address and prints it to the execution log.



Note: For details about working with a transaction's shipping or billing address, see [Scripting Transaction Shipping and Billing Addresses](#).

```
/**
 * @NApiVersion 2.x
 * @NScriptType ClientScript
 */
define([ 'N/record' ],
    function(record) {
        function pageInit() {
            // Load the record.

            var rec = record.load({
                type: record.Type.SALES_ORDER,
                id: 5025,
                isDynamic: true
            });

            // Retrieve the subrecord.

            var subrec = rec.getSubrecord({
                fieldId: 'shippingaddress'
            });

            // Create a variable and initialize it to the
            // value of the subrecord's city field.

            var cityValue = subrec.getValue({
                fieldId: 'city'
            });

            // Print the value to the execution log.

            try {
                log.debug({
                    title: 'city value',
                    details: 'city value: ' + cityValue
                });
            } catch (e) {
                log.error({
                    title: e.name,
                    details: e.message
                });
            }
        }
    }
);
```

```
        }
    }

    return {
        pageInit: pageInit
    }
});
```