



SuiteScript 2.0 API

April 6, 2016
Version 2016 Release 1

General Notices

Sample Code

NetSuite Inc. 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.

NetSuite 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, customers may not use the Service in excess of limits or thresholds that NetSuite considers commercially reasonable for the Service. If NetSuite reasonably concludes that a customer’s use is excessive and/or will cause immediate or ongoing performance issues for one or more of NetSuite’s other customers, NetSuite may slow down or throttle such customer’s excess use until such time that the customer’s use stays within reasonable limits. If a 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.

Integration with Third Party Applications

NetSuite 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 NetSuite access to Customer’s account(s) on such third party applications. NetSuite 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 NetSuite.

Copyright

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

© 2016 NetSuite Inc.

Table of Contents

1. 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
2. SuiteScript 2.0 Script Types and Entry Points	6
Bundle Installation Script Type	10
Bundle Installation Script Entry Points	11
Client Script Type	13
Client Script Entry Points	20
Map/Reduce Script Type	26
Map/Reduce Stages	31
Map/Reduce Script Governance	33
Check the Status of a Map/Reduce Script	34
Map/Reduce Script API	36
Mass Update Script Type	66
Mass Update Script Entry Points	66
Portlet Script Type	67
render(params)	69
RESTlet Script Type	70
RESTlet Script Entry Points	71
Scheduled Script Type	76
Scheduled Script Entry Points	77
Scheduled Script API	77
Suitelet Script Type	78
onRequest(params)	81
User Event Script Type	82
User Event Script API	86
Workflow Action Script Type	89
onAction(scriptContext)	91
3. SuiteScript 2.0 Global Objects and Methods	93
Module Dependency Paths to Custom Modules	93
define object	95
define([dependencies,] callback)	96
define(moduleobject)	98
require object	99
require([dependencies,] callback)	100
log Object	101
util Object	102
toString()	102
JSON object	103

JSON.parse(text)	103
JSON.stringify(obj)	104
Promise object	105
4. SuiteScript 2.0 Modules	108
N/auth Module	109
auth.changeEmail(options)	110
auth.changePassword(options)	111
N/config Module	112
config.load(options)	114
config.Type	115
N/crypto Module	116
crypto.Cipher	119
crypto.CipherPayload	120
crypto.Decipher	121
crypto.Hash	122
crypto.Hmac	124
crypto.SecretKey	125
crypto.createCipher(options)	126
crypto.createDecipher(options)	127
crypto.createHash(options)	127
crypto.createHmac(options)	128
crypto.createSecretKey(options)	128
crypto.EncryptionAlg	129
crypto.HashAlg	129
crypto.Padding	130
N/currency Module	130
currency.exchangeRate(options)	131
N/email Module	133
email.send(options)	134
email.send.promise(options)	138
email.sendBulk(options)	138
email.sendBulk.promise(options)	142
email.sendCampaignEvent(options)	142
email.sendCampaignEvent.promise(options)	143
N/encode Module	143
encode.convert(options)	144
encode.Encoding	145
N/error Module	146
error.SuiteScriptError	148
error.UserEventError	150
error.create(options)	154
N/file Module	155
file.File	157
file.create(options)	166

file.delete(options)	167
file.load(options)	168
file.Encoding	169
file.Type	170
N/format Module	171
format.format(options)	172
format.parse(options)	174
format.Type	175
format.Timezone	176
N/http Module	179
http.ClientResponse	183
http.ServerRequest	185
http.ServerResponse	188
http.get(options)	195
http.get.promise(options)	195
http.delete(options)	196
http.delete.promise(options)	197
http.request(options)	197
http.request.promise(options)	198
http.post(options)	198
http.post.promise(options)	199
http.put(options)	199
http.put.promise(options)	200
http.CacheDuration	200
http.Method	201
N/https Module	201
https.SecureString	206
https.createSecureKey(options)	209
https.createSecureKey.promise(options)	209
https.createSecureString(options)	210
https.createSecureString.promise(options)	210
https.ClientResponse	211
https.ServerRequest	213
https.ServerResponse	216
https.get(options)	223
https.delete(options)	224
https.request(options)	224
https.post(options)	225
https.put(options)	226
https.CacheDuration	227
https.Method	227
N/log Module	228
log.audit(options)	230
log.debug(options)	231

log.emergency(options)	232
log.error(options)	233
N/plugin Module	234
plugin.findImplementations(options)	235
plugin.loadImplementation(options)	236
N/portlet Module	236
portlet.resize	238
portlet.refresh	239
N/record Module	239
record.Field	254
record.Record	260
record.attach(options)	289
record.attach.promise(options)	290
record.copy(options)	291
record.copy.promise(options)	293
record.create(options)	293
record.create.promise(options)	296
record.delete(options)	296
record.delete.promise(options)	297
record.detach(options)	298
record.detach.promise(options)	299
record.load(options)	299
record.load.promise(options)	301
record.submitFields(options)	302
record.submitFields.promise(options)	303
record.transform(options)	304
record.transform.promise(options)	308
record.Type	309
N/redirect Module	311
redirect.redirect(options)	312
redirect.toRecord(options)	313
redirect.toSavedSearch(options)	314
redirect.toSavedSearchResult(options)	315
redirect.toSearch(options)	315
redirect.toSearchResult(options)	316
redirect.toSuitelet(options)	316
redirect.toTaskLink(options)	317
N/render Module	318
render.EmailMergeResult	321
render.TemplateRenderer	322
render.bom(options)	328
render.create()	329
render.mergeEmail(options)	329
render.packingSlip(options)	330

render.pickingTicket(options)	331
render.statement(options)	332
render.transaction(options)	333
render.xmlToPdf(options)	334
render.PrintMode	335
N/runtime Module	335
runtime.Script	339
runtime.Session	343
runtime.User	345
runtime.getCurrentScript()	350
runtime.getCurrentSession()	351
runtime.getCurrentUser()	351
runtime.isFeatureInEffect(options)	352
runtime.accountId	353
runtime.envType	353
runtime.executionContext	353
runtime.queueCount	354
runtime.version	354
runtime.ContextType	355
runtime.EnvType	355
runtime.Permission	356
N/search Module	356
search.Search	365
search.Result	374
search.Column	377
search.Filter	381
search.ResultSet	384
search.Page	387
search.PagedData	392
search.PageRange	395
search.create(options)	396
search.create.promise(options)	398
search.load(options)	399
search.load.promise(options)	400
search.delete(options)	401
search.delete.promise(options)	402
search.duplicates(options)	402
search.duplicates.promise(options)	404
search.global(options)	404
search.global.promise(options)	405
search.lookupFields(options)	406
search.lookupFields.promise(options)	408
search.createColumn(options)	409
search.createFilter(options)	410

search.Operator	411
search.Sort	412
search.Summary	413
search.Type	413
N/sso Module	415
sso.generateSuiteSignOnToken(options)	416
N/task Module	417
task.ScheduledScriptTask	426
task.ScheduledScriptTaskStatus	428
task.MapReduceScriptTask	430
task.MapReduceScriptTaskStatus	433
task.CsvImportTask	441
task.CsvImportTaskStatus	445
task.EntityDeduplicationTask	446
task.EntityDeduplicationTaskStatus	450
task.WorkflowTriggerTask	451
task.WorkflowTriggerTaskStatus	454
task.create(options)	455
task.checkStatus(options)	456
task.TaskType	456
task.TaskStatus	457
task.MasterSelectionMode	458
task.DedupeMode	459
task.DedupeEntityType	459
task.MapReduceStage	460
N/transaction Module	461
transaction void(options)	462
transaction void.promise(options)	463
transaction.Type	464
N/ui/dialog Module	465
dialog.alert(options)	467
dialog.confirm(options)	468
dialog.create(options)	469
N/ui/message module	470
message.Message	471
message.create(options)	472
message.Type	473
N/ui/serverWidget Module	474
serverWidget.Assistant	484
serverWidget.AssistantStep	504
serverWidget.Button	511
serverWidget.Field	513
serverWidget.FieldGroup	527
serverWidget.Form	530

serverWidget.List	553
serverWidget.ListColumn	560
serverWidget.Sublist	563
serverWidget.Tab	572
serverWidget.createAssistant(options)	574
serverWidget.createForm(options)	575
serverWidget.createList(options)	576
serverWidget.AssistantSubmitAction	576
serverWidget.FieldBreakType	577
serverWidget.FieldDisplayType	578
serverWidget.FieldLayoutType	579
serverWidgetFieldType	580
serverWidget.FormPageLinkType	581
serverWidget.LayoutJustification	581
serverWidget.ListStyle	582
serverWidget.SublistDisplayType	583
serverWidget.SublistType	583
N/url Module	584
url.format(options)	585
url.resolveRecord(options)	586
url.resolveScript(options)	587
url.resolveTaskLink(options)	588
N/util Module	589
util.isArray(obj)	590
util.isBoolean(obj)	591
util.isDate(obj)	592
utilisFunction(obj)	592
util.isNumber(obj)	593
util.isObject(obj)	593
util.isRegExp(obj)	594
util.isString(obj)	595
util.nanoTime()	595
util.each(iterable, callback)	596
util.extend(receiver, contributor)	597
N/workflow Module	598
workflow.initiate(options)	599
workflow.trigger(options)	601
N/xml Module	602
xml.Parser	609
xml.XPath	611
xml.Node	612
xml.Document	633
xml.Element	649
xml.Attr	664

xml.escape(options)	667
xml.validate(options)	667
xml.NodeType	669
5. SuiteScript 2.0 JSDoc Validation	670
Controlling Access to Scripts and Custom Modules	673
6. SuiteScript 2.0 Entry Point Script Deployment	676
Record-Level and Form-Level Scripts	676
SuiteScript 2.0 Entry Point Script Validation	677
Entry Point Script Validation Guidelines	677
Entry Point Script Validation Examples	680
Entry Point Script Validation Error Reference	683
SuiteScript 2.0 Record-Level Scripts	687
Script Record Creation	687
Script Deployment	691
SuiteScript 2.0 Form-Level Scripts	694
Attaching a Client Script to a Form	694
Configuring a Custom Action	696

Chapter 1 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
- Client-side Scriptable Cart and Checkout

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	<p>Major versions of SuiteScript include significant functionality changes and improvements.</p> <p>Major versions are not backward compatible with previously released versions.</p>
Minor	Version 3.1, 3.2, 3.3	<p>Minor versions of SuiteScript include enhancements to existing features.</p> <p>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.</p>
Patch	Does not apply	<p>Patch versions of SuiteScript are included with regular NetSuite bug fix releases.</p> <p>Patch versions are backward compatible with all versions released since the last major version.</p>

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.

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.

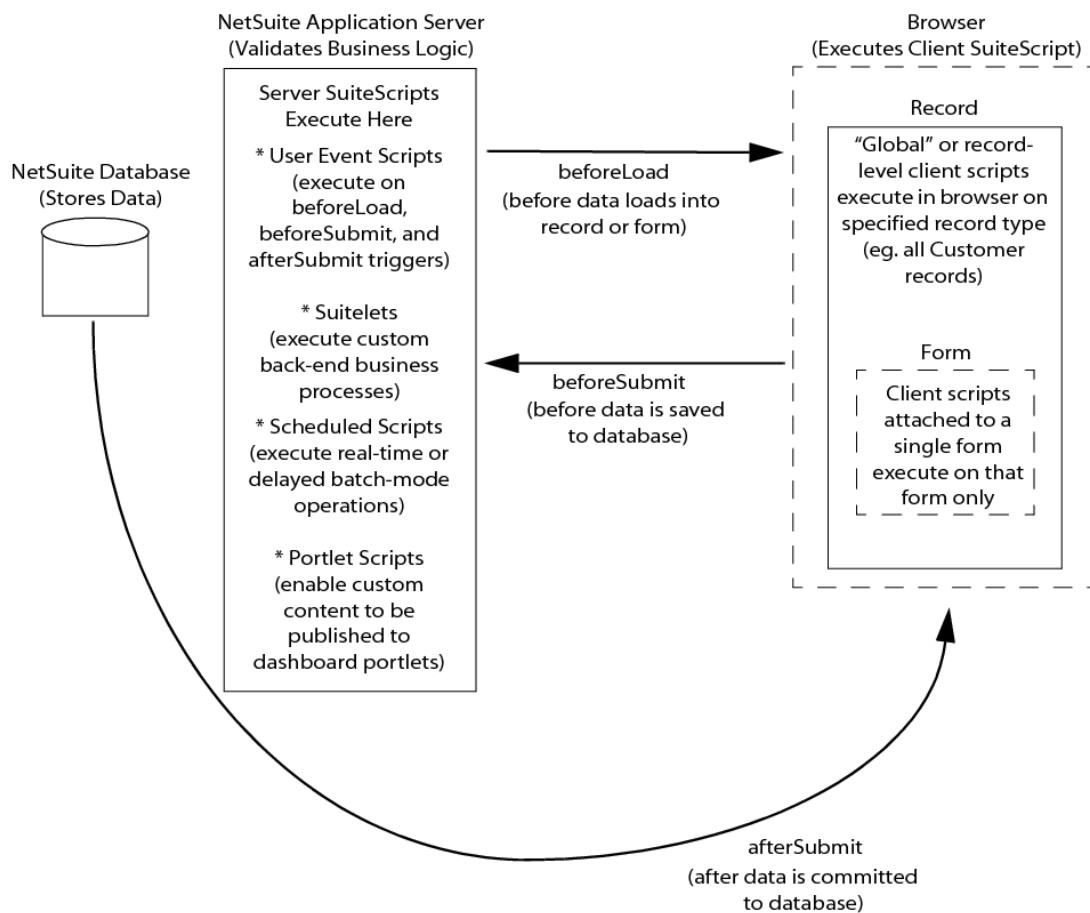
Note: Your specific process may vary, depending on the content of your script.

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

Chapter 2 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.

SuiteScript 2.0 Script Type Entry Points

Script entry points are included in the APIs. Each entry point script (the script attached to the script record) must include at least one entry point and corresponding entry point function. An entry point function is a public function that is executed when a script entry point is triggered. The primary logic within an entry point script is contained within its entry point functions.

Important: Place all logic in at least one named function. In an entry point script, your logic must be executed after an entry point is invoked. Logic executed before an entry point is invoked throws an error.

```
/**  
 * @NApiVersion 2.x  
 */  
define(['N/record'],  
    function(record) {  
        function alertPageLoaded(context) {  
            alert('Page is loaded');  
        };  
        function fieldChanged(context) {  
            var currentRecord = context.currentRecord;  
            auditRecordChanges(currentRecord.type, currentRecord.id);  
            updateMemoFieldWhenItemFieldIsChanged(context);  
        }  
        function auditRecordChanges(type, id) {  
            var logRecord = record.create({type: 'customrecord_log', isDynamic: true});  
            logRecord.setValue('custrecord_recordType', type);  
            logRecord.setValue('custrecord_recordId', id);  
            logRecord.setValue('custrecord_changedDate', new Date());  
            logRecord.save();  
        }  
        function updateMemoFieldWhenItemFieldIsChanged(context) {  
            if(context.fieldId === 'item') {  
                context.currentRecord.setValue('memo', 'item field is changed');  
            }  
        }  
        return({  
            pageInit: alertPageLoaded,  
            fieldChanged: fieldChanged  
        });  
    });
```

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.
- Use **static** ID values in your API calls where applicable because name values can be changed.
- Use custom name spaces or unique prefixes for all your function names.
- When working with script type events, your function name should correspond with the event. For example, a pageInit event can be named PageInit 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 but and assists NetSuite support in locating problems if necessary.
- You must use the **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 Work Orders and Multiple Currencies features are enabled in the target NetSuite account, and that the bundle that is being installed is version 2.0.
- After the bundle installation and the bundle update, create an account record in the target account if the update changed the bundle version number.

Note: Accounts are not available to be included in bundles.

```
/**  
 * @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) {  
            this.checkPrerequisites();  
        },  
        beforeUpdate: function beforeUpdate(params) {  
            this.checkPrerequisites();  
        }  
    };  
});
```

```

        this.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
params.version	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
params.fromVersion	number	The version of the bundle that is currently installed in the target account.	Version 2016 Release 1

Parameter	Type	Description	Since
params.toVersion	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
params.version	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

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

Parameter	Type	Description	Since
params.version	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

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 be used to validate user-entered data and to 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.

Important: SuiteScript 2.0 client scripts are not currently supported with Scriptable Cart and Checkout

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
- Searching for a record
- Loading, saving or deleting a record

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 an example of a client remote object script. On the saveRecord client event, the script uses the currentrecord module to create a new estimate record. This script creates the new Estimate, sets values on the new record, and then submits the record, all from within a script that is executed on the client.

```
/*
 * @NApiVersion 2.x
 * @NScriptType ClientScript
 */
define(['N/record'],
    function(record) {

        function saveRec() {
            // access the NetSuite server to
            // instantiate a new Estimate
            var rec = record.create({
                type: 'estimate',
                isDynamic: true
            });

            rec.setValue({
                fieldId: 'entity',
                value: '107'
            });

            rec.insertLine({
                sublistId: 'item',
                value: 0
            });

            rec.setCurrentSublistValue({
                sublistId: 'item',
                fieldId: 'item',
                value: 244
            });

            rec.setCurrentSublistValue({
                sublistId: 'item',
                fieldId: 'quantity',
                value: 1
            });

            rec.setCurrentSublistValue({
                sublistId: 'item',
                fieldId: 'amount',
                value: 24
            });

            rec.commitLine({
                sublistId: 'item'
            });

            var id = rec.save();
        }

        saveRec();
    }
);
```

```
        return id > -1;
    }

    return {
        saveRecord: saveRec
    }
});
```

Role Restrictions

Client SuiteScript respects 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 is a client script, which you can attach to a custom sales order form and set to execute on the field change client event:

```
/**
 * @NApiVersion 2.x
 * @NScriptType ClientScript
 */
define(['N/record', 'N/search'],
    function(record, search) {
        function getSalesRepEmail() {
            var salesRep = record.getValue({
                fieldId: 'salesrep'
            });
            var salesRepEmail = search.lookupFields({
                type: 'employee',
                id: salesRep
                columns: ['email']
            });
            alert(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.
- The execution of `record.setValue` and `record.setCurrentSublistValue` is multi-threaded whenever child field values need to be sourced in. Use the `postSourcing` function to synchronize your logic. Setting the synchronous parameter, `fireSlavingSync`, to true accomplishes the same thing.
- Use `record.setValue` and `record.setCurrentSublistValue` instead of `record.setText` and `record.setCurrentSublistText` if the field that you are setting renders as a popup text field. Your script executes more predictably using the `xxxValue` functions because you are setting an internal ID — a precise definition. The `xxxText` functions perform searches that may return duplicates or no results.

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: 'pricelvels'
            }) !== '1-1')
                currentRecord.setCurrentSublistValue({
                    sublistId: sublistName,
                    fieldId: 'pricelvels',
                    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 or editor sublist.

fieldChanged

Description	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: <ul style="list-style-type: none">• Provide the user with additional information based on user input.• Disable or enable fields based on user input. 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 .
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	string	The current form record.	Version 2015 Release 2
scriptContext.sublistId	string	The sublist ID name.	Version 2015 Release 2

Parameter	Type	Description	Since
scriptContext.fieldId	string	The field ID name.	Version 2015 Release 2
scriptContext.lineNum	string	The line number 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.columnNum	string	The column number 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 pagelinit event for line items in an inlineeditor or editor sublist.
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	string	The current form record.	Version 2015 Release 2
scriptContext.sublistId	string	The sublist ID name.	Version 2015 Release 2

pagelinit

Description	Executed when the page completes loading or when the form is reset. The following sample tasks can be performed: <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.
--------------------	---

	<ul style="list-style-type: none"> Validate that fields required for your custom code (but not necessarily required for the form) exist.
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	string	The current form record.	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

postSourcing

Description	<p>Executed on transaction forms when a field that sources information from another field is modified.</p> <p>This event behaves like a fieldChanged event once 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.</p> <p>Note: The event is not triggered by field changes for a field that does not have any slaved fields.</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	string	The current form record.	Version 2015 Release 2
scriptContext.sublistId	string	The sublist ID name.	Version 2015 Release 2
scriptContext.fieldId	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.
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>	string	The current form record.	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.
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>	string	The current form record.	Version 2015 Release 2
<code>scriptContext.sublistId</code>	string	The sublist ID name.	Version 2015 Release 2

validateDelete

Description	Executed when removing an existing line from an edit sublist.
--------------------	---

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>	string	The current form record.	Version 2015 Release 2
<code>scriptContext.sublistId</code>	string	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. Note: This event does not apply to drop-down select or check box fields.
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>	string	The current form record.	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
<code>scriptContext.lineNum</code>	string	The line number if the field is in a sublist or a matrix.	Version 2015 Release 2

Parameter	Type	Description	Since
		If the field is not a sublist or matrix, the default value is undefined.	
scriptContext.columnNum	string	The column number 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.
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
<code>scriptContext.currentRecord</code>	string	The current form record.	Version 2015 Release 2
<code>scriptContext.sublistId</code>	string	The sublist ID name.	Version 2015 Release 2

validateLine

Description	Executed before a line is added to an inlineeditor or editor sublist. This event can behave like a <code>saveRecord</code> event for line items in an inlineeditor and editor sublist.
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>	string	The current form record.	Version 2015 Release 2
<code>scriptContext.sublistId</code>	string	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 API](#)

Map/reduce scripts provide a structured framework for server-side scripts that process a large number of records.

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 can be re-scheduled and re-queued as needed without manual intervention.

In addition, SuiteCloud Plus users can also use map/reduce scripts to process records in parallel across multiple work queues. Users manually select the work queues to utilize from the script deployment record.

You can think of map/reduce scripts as a subclass of scheduled script. Deployment is handled in the same manner. Both script types have three options for deployment: by schedule, from the **Save and Execute** option on the deployment record, or through the task module (see [task.MapReduceScriptTask](#)).

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

Map/Reduce Script Sample

- [Example 1](#)

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 + ' Number of Queues', 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.

For more information, see [Map/Reduce Stages](#).

```
/*
 * @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

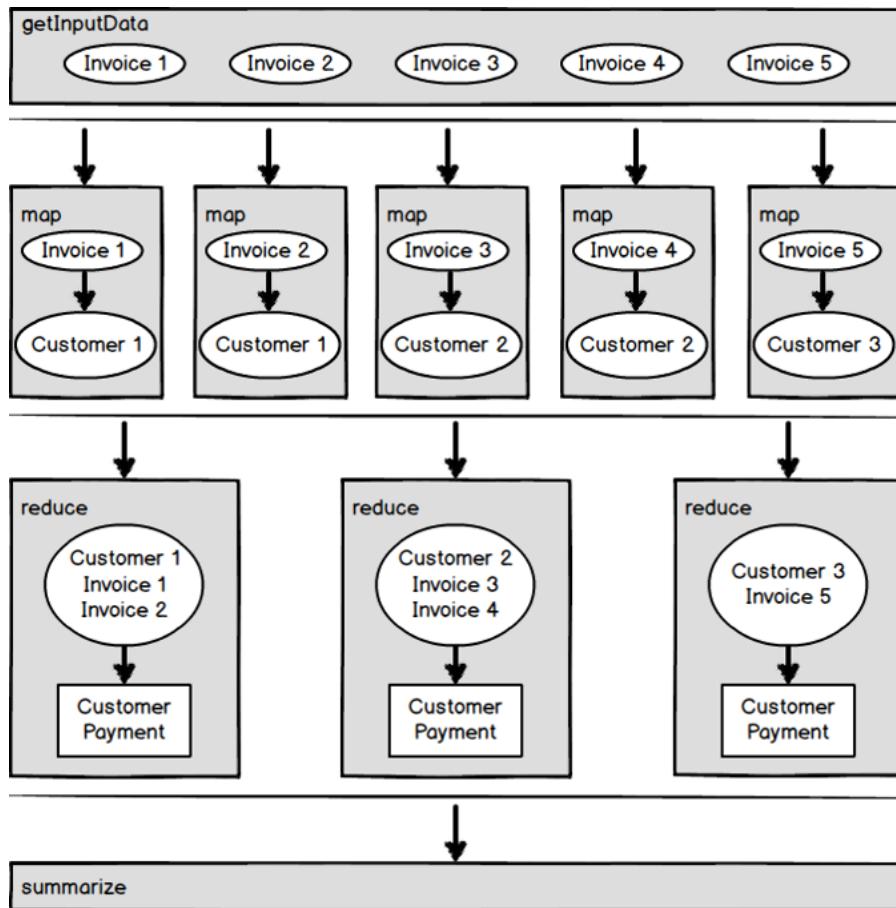
The map/reduce script type goes through at least two of five possible stages.

Important: It is not mandatory to use both the map stage and the reduce stage. You may skip one of these 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.

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/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.

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.

Map/Reduce Script Status									
Refresh									
FILTERS									
<input type="checkbox"/> SHOW INACTIVES									
SCRIPT	DEPLOYMENT ID	DATE CREATED	START DATE	END DATE	STATUS	STAGE	QUEUES	DETAILS	CANCEL
map_reduce_sample	customdeploy1	11/19/2015 10:33:41 am	10:42 am		Failed		1	Details	
map_reduce_sample	customdeploy1	11/19/2015 10:47:58 am	10:47 am	10:52 am	Complete		2	Details	

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

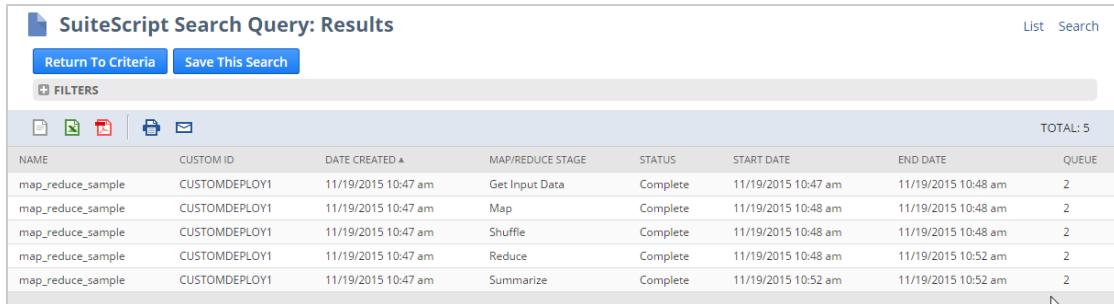
If you are a SuiteCloud Plus user, reviewing these details can help you to isolate a problem to a particular queue. Use the links on the script status page to jump back to the deployment record and edit queue assignments.

View Details of Script Instances

A script instance is the individual job (work queue item) associated with a task in the queue.

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.



The screenshot shows a search results page titled "SuiteScript Search Query: Results". At the top, there are buttons for "Return To Criteria" and "Save This Search", and a "FILTERS" button. Below the header is a toolbar with icons for file operations like Open, Save, Print, and Email. A "TOTAL: 5" label is in the top right corner. The main area is a table with the following columns: NAME, CUSTOM ID, DATE CREATED, MAP/REDUCE STAGE, STATUS, START DATE, END DATE, and QUEUE. The data rows are as follows:

NAME	CUSTOM ID	DATE CREATED	MAP/REDUCE STAGE	STATUS	START DATE	END DATE	QUEUE
map_reduce_sample	CUSTOMDEPLOY1	11/19/2015 10:47 am	Get Input Data	Complete	11/19/2015 10:47 am	11/19/2015 10:48 am	2
map_reduce_sample	CUSTOMDEPLOY1	11/19/2015 10:47 am	Map	Complete	11/19/2015 10:48 am	11/19/2015 10:48 am	2
map_reduce_sample	CUSTOMDEPLOY1	11/19/2015 10:47 am	Shuffle	Complete	11/19/2015 10:48 am	11/19/2015 10:48 am	2
map_reduce_sample	CUSTOMDEPLOY1	11/19/2015 10:47 am	Reduce	Complete	11/19/2015 10:48 am	11/19/2015 10:52 am	2
map_reduce_sample	CUSTOMDEPLOY1	11/19/2015 10:47 am	Summarize	Complete	11/19/2015 10:52 am	11/19/2015 10:52 am	2

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 queue 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..

Script Deployment

Actions ▾

SCRIPT map_reduce_sample	LOG LEVEL Debug
TITLE * map_reduce_sample	EXECUTE AS ROLE Administrator
ID customdeploy1	QUEUES * <input type="checkbox"/> Select All
<input checked="" type="checkbox"/> DEPLOYED	1
STATUS *	2
Not Scheduled	3
SEE INSTANCES Status Page	4
YIELD AFTER MINUTES *	
60	

Schedule • Execution Log History •

SINGLE EVENT
 DAILY EVENT
 WEEKLY EVENT
 MONTHLY EVENT
 YEARLY EVENT

START DATE * 11/19/2015 START TIME 6:00 pm REPEAT

END BY 11/19/2015 NO END DATE

Actions ▾

Map/Reduce Script API

- *Map/Reduce Script Entry Points*
- 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 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 generate the initial data collection for the first stage. You do not directly provide input data (contained in the MapContext, ReduceContext, or Summary Objects) to the remaining script entry points. This is handled by the framework based on the original input data you provide.

<code>getInputData()</code>	Marks the beginning of the map/reduce process. Invokes the input stage. This entry point is required.
<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.

Map/Reduce Object Members

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

Member Type	Name	Description
		Passed in when <code>reduce(reduceContext)</code> is called.
	<code>mapReduce.ReduceSummary</code>	Holds statistics regarding the reduce stage.
	<code>mapReduce.Summary</code>	Holds statistics regarding the map/reduce process.

GetInputContext Object Members

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

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

Iterator Object Members

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

Member Type	Name	Return Type	Description
Method	<code>Iterator.iterator().each(iteratorFunction)</code>	void	Takes a developer-defined function. This function is called once 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.
	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 number of queues used at the same time while 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 while running <code>map(mapContext)</code> .
	MapSummary.usage	number	Total number of usage units consumed while running <code>map(mapContext)</code> .
	MapSummary.yields	number	Total number of times yielding the queue 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.

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

getInputData()

Description	<p>Executes when the <code>getInputData</code> entry point is triggered.</p> <p>Marks the beginning of the map/reduce process. The purpose of the input stage is to generate the input data. This entry point is required.</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>
Returns	<p><code>Array</code> <code>Object</code> <code>search.Search</code> <code>mapReduce.ObjectRef</code></p> <p>Examples:</p> <ul style="list-style-type: none"> • Array <pre>[{a : 'b'}, {c : 'd'}]</pre>

	<ul style="list-style-type: none"> • Object <pre>{ a: {...}, b: {...} }</pre> <ul style="list-style-type: none"> • search.Search Object <pre>search.load(1234)</pre> <ul style="list-style-type: none"> • Object Reference <pre>{ type: 'search', id: 1234 }</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.</p> <p>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.</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_REGEXP.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 `amapReduce.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 + ' Number of Queues', 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;
    });
...

```

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 the getInputData() function was stopped and invoked again. If the Java virtual machine (JVM) restarts, to reduce negative impact to map/reduce processing, NetSuite automatically restarts the current map/reduce function. Any data previously written by the incomplete function is deleted. If the value is <code>true</code> , the current process invoked by getInputData() was restarted.
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 type is 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
}
...
```

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"
Important:	This is the only supported value.
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 map(mapContext) 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/reduce function. Any data previously written by the incomplete function is deleted. If the value is <code>true</code> , the current process invoked by map(mapContext) was restarted.
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. Note: Each key cannot exceed 4000 bytes.
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 search.Result object converted to a JSON string with <code>JSON.stringify()</code>. Note: Each value cannot exceed 1 megabyte.
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_REGEXP.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 map/reduce function. Any 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.
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	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: <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.
-----------------------------	---

Note: Each key cannot exceed 4000 bytes.

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	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: <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	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.write(key, value)

Method Description	Writes the key/values groups.
---------------------------	-------------------------------

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 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 once 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. Note: Each value cannot exceed 1 megabyte.

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 + ' Number of Queues', 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 while 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 while 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 number of queues that executed tasks simultaneously for the map/reduce script. Note: This number can be less than the number of queues allocated on the script deployment. For example, a queue allocated with tasks that remained in the pending state is 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(type + ' Number of Queues', summary.concurrency);  
...
```

Summary.yields

Property Description	Total number of times yielding the queue while 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 map/reduce function. Any 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.
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 getInputData() 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 while running getInputData() (does not include idle time in the queue).
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 while 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 while 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 while 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 number of queues that executed tasks simultaneously for the map stage.
Note:	This number can be less than the number of queues allocated on the script deployment. For example, a queue allocated with tasks that remained in the pending state is 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 yielding the queue while 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 <code>map(mapContext)</code> function.
Type	<code>mapReduce.Iterator</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](#).

```
...
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 <code>map(mapContext)</code> .
Type	<code>mapReduce.Iterator</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](#).

```
...
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 while 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 while running reduce(reduceContext) .
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 number of queues that executed tasks simultaneously for the reduce stage.
	Note: This number can be less than the number of queues allocated on the script deployment. For example, a queue allocated with tasks that remained in the pending state is 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 yielding the queue while running reduce(reduceContext) .
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%:

```

/**
 * @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 can be created:

- **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.

Portlet scripts can only run after users have added the scripts to their dashboards. Once the scripts have been added, users must then open their dashboards for a portlet script to execute.

Form Portlet Script Sample

```
/**
 *@NApiVersion 2.x
 *@NScriptType Portlet
 */
define(['N/search'],
    function(search) {
        function render(params) {
            var portlet = params.portlet;
            portlet.setTitle('Simple Form Portlet');
            var fld = portlet.addField('text', 'text', 'Text');
            fld.setLayoutType('normal', 'startcol');
            portlet.setSubmitButton('http://httpbin.org/post', 'Submit', '_top');
        }
        return {

```

```
        render: render
    );
});
```

HTML Portlet Script Sample

```
/**
 * @NApiVersion 2.x
 * @NScriptType Portlet
 */
define([], function() {
    function render(params) {
        params.portlet.setTitle('My Portlet');
        var content = '<td><span><b>Hello!!!</b></span></td>';
        params.portlet.setHtml(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.setTitle('Search Engines');
            portlet.addLine('Google', 'http://www.google.com/');
            portlet.addLine('Bing', 'http://www.bing.com/');
        }
        return {
            render: render
        };
});
```

List Portlet Script Sample

```
/**
 * @NApiVersion 2.x
 * @NScriptType Portlet
 */
define(['N/search'],
    function(search) {
        function render(params) {
            var isDetail = (params.column === 2);
            var portlet = params.portlet;
```

```

portlet.setTitle(isDetail ? "My Detailed List" : "My List");
portlet.addColumn('internalid', 'text', 'Number', 'LEFT');
portlet.addColumn('entityid', 'text', 'ID', 'LEFT');
if (isDetail) {
    portlet.addColumn('email', 'text', 'E-mail', 'LEFT');
    portlet.addColumn('custentity_multiselect', 'text', 'Multiselect', '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 params parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
params.portlet	Portlet	required	The portlet object used for rendering.	Version 2015 Release 2
params.column	integer	required	The column index for the portlet on the dashboard. Use one of the following numeric values: 1. left column 2. center column 3. right column	Version 2015 Release 2
params.entityid	string	required for custom portlets	The customer ID for the selected customer.	Version 2015 Release 2

RESTlet Script Type

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

```

RESTlet Script Entry Points

Script Entry Point	
get	Returns the HTTP response body.
delete	Returns the HTTP response body.
put	Returns the HTTP response body.
post	Returns the HTTP response body.

delete

Description	Returns the HTTP response body. <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

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	Returns the HTTP response body. <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

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

Parameter	Type	Description	Since
requestParams	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	Returns the HTTP response body. <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

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
requestBody	string Object	required	The HTTP request body. <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	Returns the HTTP response body. <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

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

Tracking and Managing RESTlet Activity

If you use token-based authentication, you have the ability to track calls that were made by external applications to RESTlets hosted in your NetSuite account. You can track RESTlet activity by using integration records.

When using token-based authentication (TBA), you create an integration record to represent each external application that calls RESTlets or sends web services requests. You use the integration record to generate the consumer key and secret needed by the application to authenticate. You can also use the record to do the following:

- Block requests that reference the record's consumer key. You can block requests by setting the record's State field to Blocked.
- Enable or disable token-based authentication for an external application. If the application also makes web services requests, an additional authentication option (User Credentials) is available. Checking or clearing the User Credentials box has no effect on whether the application can call RESTlets. The User Credentials option affects only web services calls.

Integration records are located at Setup > Integration > Manage Integrations. The record can be accessed only by administrative users.

For full details on using the integration record to set up token-based authentication, see the help topic [Token-based Authentication](#).

For more information on using integration records in conjunction with RESTlets, see the following topics:

- Ownership of Integration Records

- [Using the RESTlets Execution Log](#)
- [Finding System Notes for an Integration Record](#)
- [More Information](#)

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 editable 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.

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.

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

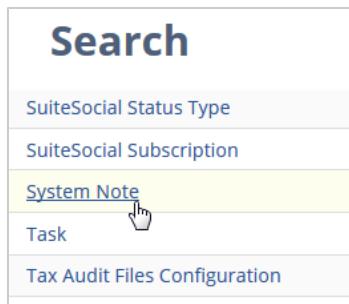
record, the initial values of its fields, and subsequent updates. For example, if a user changed the State field from Blocked to Enabled, a system note would provide a record of that change.

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](#).

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	DATE	SET BY	CONTEXT	TYPE	FIELD	OLD VALUE	NEW VALUE
- All -	Default	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	

More Information

For more information about managing integration records, see the help topic [Managing Integrations](#), which is part of the SuiteTalk (Web Services) Platform Guide. Some of the detail in this guide pertains only or primarily to web services, but the following sections may be of interest:

- [Adding an Integration Record](#)
- [Regenerating a Consumer Key and Secret](#)
- [Distributing by Bundling](#)

- Using Integration Records in Sandbox Accounts
- Removing Integration Records
- Special Cases Related to System Notes Logging

Scheduled Script Type

```
/**  
 * @NApiVersion 2.x  
 * @NScriptType ScheduledScript  
 */  
define(['N/search', 'N/record', 'N/email', 'N/runtime'],  
    function(search, record, email, runtime) {  
        // Runs a search on the sales orders created today and fulfill them.  
        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('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 Entry Points

Script Entry Point	
execute	Definition of the scheduled script trigger point

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. It is automatically passed to the script entry point by NetSuite

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

Scheduled Script API

API	
<code>context.InvocationType</code>	Enumeration that holds the string values for scheduled script execution contexts.

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

- ON_DEMAND
- USER_INTERFACE
- ABORTED
- SKIPPED

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, sublists, 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:

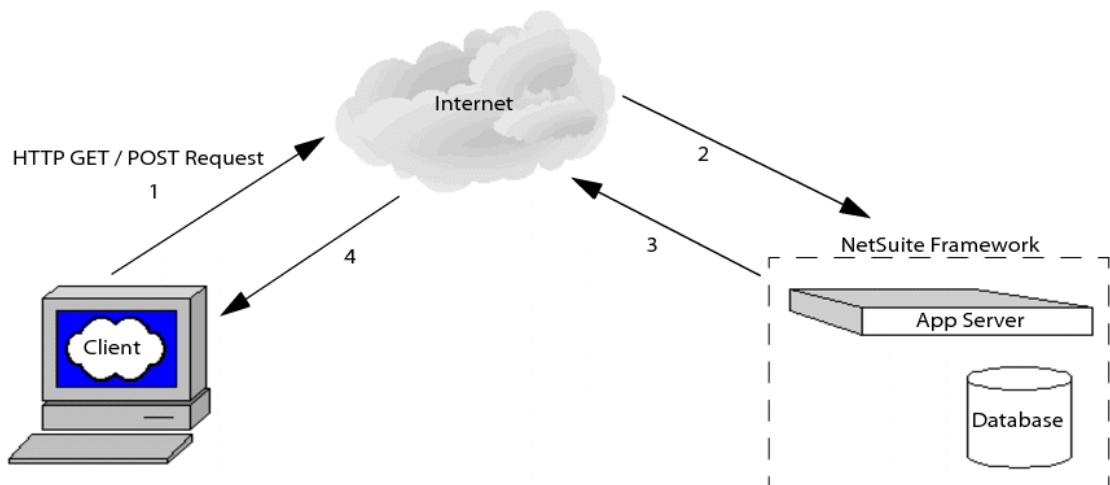
1. Client initiates an HTTP GET or POST request (typically from a browser) for a system-generated URL. A web request object contains the data from the client's request.
2. The user's script is invoked, which gives the user access to the entire Server SuiteScript API as well as a web request and web response 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

Reserved Suitelet URL Parameter Names	
I	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 Setting Available Without Login.)
- Append “ifrmcntnr=T” to the external URL when embedding in iFrame especially if you are using Firefox. (See Embed a Suitelet in iFrame.)
- 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 allows you to 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

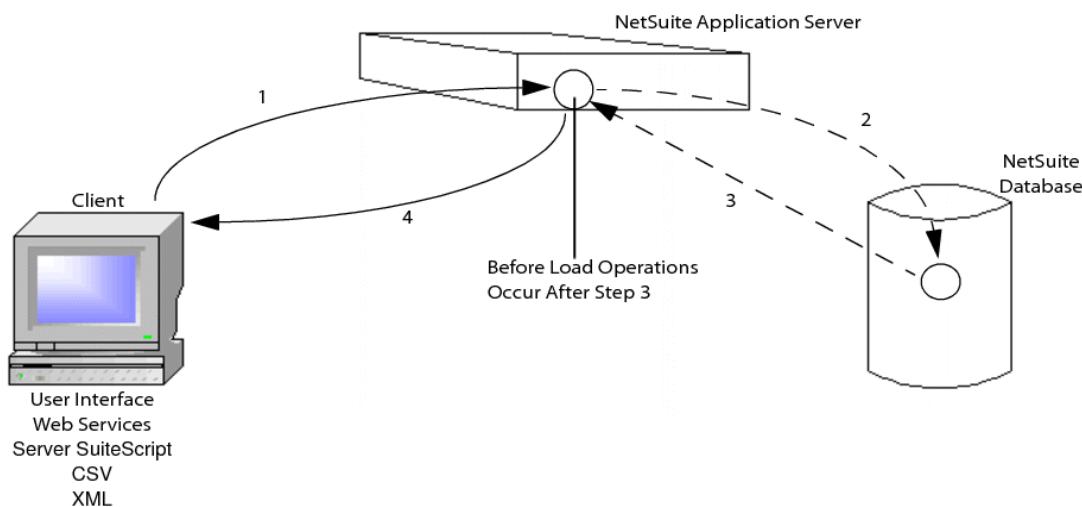
User event scripts are executed based on operation types defined as `beforeLoad`, `beforeSubmit`, and `afterSubmit`.

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 pagelinit client script for this purpose. See [pagelinit](#).



The following steps and diagram provide an overview of what occurs on beforeSubmit and afterSubmit operations:

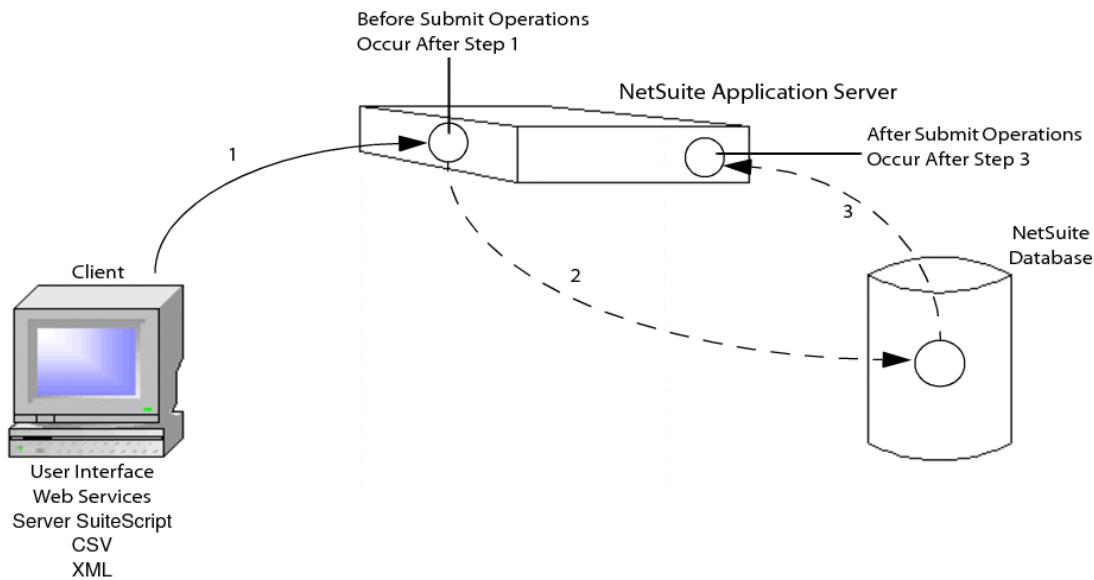
1. The client performs a write operation by submitting data to the application server. (The client request can come from the user interface, web services, server-side SuiteScript calls, CSV imports, or XML.) The application server:
 - a. performs basic permission checks on the client
 - b. processes the submitted data and performs specified validation checks during a beforeSubmit operation

The submitted data has **NOT** yet been committed to the database.

2. Once 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 in-line. 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 will also execute as a result 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 that creates a follow-up phone call record for every newly created customer record.

```
/**  
 * @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 {  
                    call.insert();  
                } catch (e) {  
                    log.error(e.message);  
                }  
            }  
        }  
    }  
);
```

```

        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. Note: Asynchronous afterSubmit user events are only supported during webstore checkout.
Returns	void
Since	Version 2015 Release 2

Parameters

Note: The scriptContext parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
scriptContext.newRecord	record	required	The new record	Version 2015 Release 2
scriptContext.oldRecord	string	required	The old record	Version 2015 Release 2
scriptContext.type	form	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 pagelnit client script for this purpose. Note: beforeLoad user events cannot be triggered when you load/access an online form.
Returns	void
Since	Version 2015 Release 2

Parameters

Note: The scriptContext parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
scriptContext.newRecord	record	required	The new record.	Version 2015 Release 2
scriptContext.type	string	required	<p>The type of operation invoked by the event.</p> <p>The type can be any of the possible values for the <code>context.UserEventType</code> enum.</p> <p>Type arguments vary depending on whether the event occurs during a beforeLoad, beforeSubmit, or afterSubmit operation.</p> <p>This parameter allows the script to branch out to different logic depending on the operation type. For example, a script with deltreen logic that deletes a record and all of its child records should only be invoked when type equals delete.</p> <p>User event scripts should always check the value of the type argument to avoid indiscriminate execution.</p>	Version 2015 Release 2
scriptContext.form	form	required	The current form.	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	void
Since	Version 2015 Release 2

Parameters

Note: The `scriptContext` parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
<code>scriptContext.newRecord</code>	record	required	The new record	Version 2015 Release 2
<code>scriptContext.oldRecord</code>	string	required	The old record	Version 2015 Release 2
<code>scriptContext.type</code>	form	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 sublists 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([ 'N/record', 'N/runtime', 'N/search' ], function(record, runtime, search) {  
    return {  
        onAction : function(scriptContext) {  
            // Get a script variable defined in the Workflow that indicates  
            // a limit on the number of expenses in an expense report  
            var stFrequentCriteria = runtime.getCurrentScript()  
                .getParameter('SCRIPT', 'custscript_sdr_freq_trav_limit');  
            var intFrequentCriteria = parseInt(stFrequentCriteria, 10);  
            // Get the entity (Employee name) of an expense report  
            var recExpenseReport = scriptContext.newRecord;  
            var stEntity = recExpenseReport.getValue('entity');  
            // Define dates for today and a month ago  
            var aMonthAgo = new Date();  
            aMonthAgo.setMonth(aMonthAgo.getMonth() - 1);  
            var today = new Date();  
            var arrSearchFilters = [ search.createFilter({  
                name : 'employee',  
                operator : search.Operator.IS,  
                values : [ stEntity ]  
            }), search.createFilter({  
                name : 'trandate',  
                operator : search.Operator.WITHIN,  
                values : [ aMonthAgo, today ]  
            }), search.createFilter({  
                name : 'mainline',  
                operator : search.Operator.IS,  
                values : [ 'T' ]  
            }) ];  
            var arrSearchColumns = [ search.createColumn({  
                name : 'internalid',  
                summary : 'count'  
            }) ];  
            var arrSearch = search.create({  
                type : 'expensereport',  
                filters : arrSearchFilters,  
                column : arrSearchColumns  
            });  
            var arrSearchResults = arrSearch.run().getRange({  
                start : 0,  
                end : 999  
            });  
            var stCount = arrSearchResults[0].getValue(arrSearchColumns[0]);  
        }  
    };  
};
```

```

var count = parseInt(stCount, 10);
var stFrequentTraveler = '';
if (count >= intFrequentCriteria) {
    stFrequentTraveler = 'Frequent';
} else {
    stFrequentTraveler = 'Infrequent';
}
var stAmount = recExpenseReport.getValue('amount');
var intExpenseLines = recExpenseReport.getLineCount('expense');
var stNotes = '';
stNotes += stFrequentTraveler + ' Traveler' + '\n';
stNotes += 'Last expense report:' + '\n';
stNotes += '--Expense report total = ' + stAmount + '\n';
stNotes += '--Expense lines = ' + intExpenseLines;
var recEmployee = record.load({
    type : 'employee',
    id : stEntity,
    isDynamic : true
});
recEmployee.setValue('comments', stNotes);
var stEmployeeId = recEmployee.save();
if (stEmployeeId === '') {
    return 'F';
} else {
    return 'T';
}
}
return ({
    beforeInstall : beforeInstall,
    afterInstall : afterInstall,
    beforeUpdate : beforeUpdate,
    afterUpdate : afterUpdate,
    beforeUninstall : beforeUninstall
}));
}

```

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
scriptContext.newRecord	record.Record	required	The new record. record.Record.save() is not permitted.	Version 2016 Release 1

Parameter	Type	Required / Optional	Description	Since
scriptContext.oldRecord	record.Record	required	The old record. <code>record.Record.save()</code> is not permitted.	Version 2016 Release 1

Chapter 3 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 object](#)
- [require object](#)
- [log Object](#)
- [util Object](#)
- [toString\(\)](#)
- [JSON object](#)
- [Promise object](#)

Module Dependency Paths to Custom Modules

To specify module paths and module names to custom modules, you can use the [define object](#) or the [require object](#).

SuiteScript supports the following types of module dependency paths:

Path Type	Description
Absolute Paths	Specifies the path from the root folder in the file cabinet. For example, "SuiteScripts/MyApp/Util".
Relative Paths	Specifies the path relative to the dependent module's location in the file cabinet. For example, the path for a sibling file under "SuiteScripts/MyApp" would be "./Util".
Virtual Paths	Specifies a path that is independent of the file cabinet location.

In the following example, SuiteScript 2.0 expects the custom module files lib1.js and lib2.js to be located in the same File Cabinet directory as myModule.js.

```
// myModule.js
define(['./lib1', './lib2'], // myModule has a dependency on modules lib1 and lib2
      ...
);
```

Note: Module names must be provided as valid file cabinet paths, minus the .JS file extension. If the path is invalid, the MODULE_DOES_NOT_EXIST error is thrown. For more information, see [Errors](#).

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, relative to the File Cabinet directory that contains myModule.js.

```
// myModule.js
define(['./lib/lib1', './lib/lib2'], // myModule has a dependency on modules lib1 and lib2
      ...
);
```

Virtual Paths

Use a virtual path to point to a file cabinet location that is likely to change. A virtual path is designed to be reliable even when a module is moved or the bundle id changes due to deprecation or copying. SuiteScript 2.0 checks the deprecation or copy chain and looks for the file in multiple places as needed.

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
       ...
);
```

define object

The define object is a global object that implements a define() Module Loader interface for SuiteScript 2.0. It conforms to the Asynchronous Module Definition (AMD) specification. The define() Function works as a factory function for SuiteScript 2.0 script types and custom SuiteScript 2.0 modules.

The define() Function has two signatures: [define\(\[dependencies,\] callback\)](#) and [define\(moduleobject\)](#).

When NetSuite executes a script that uses the `define([dependencies,] callback)` signature, it loads the dependencies and then executes the callback function. When NetSuite executes a script with the `define(moduleobject)` signature, it returns the module object defined by the `moduleobject` function parameter.

Use the define() Function for the following purposes in SuiteScript 2.0:

- 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. Create a custom module and return the custom module object. You can then include the custom module as dependency in another script. You can use the [define\(\[dependencies,\] callback\)](#) signature or, if the custom module does not require any dependencies, the [define\(moduleobject\)](#) signature.

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. As a result, 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.

define([dependencies,] callback)

Method Description	Function used to define a SuiteScript 2.0 script type 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. If you use this function to define a SuiteScript 2.0 script type, the return statement must return at least one entry point and entry point function. Note: This function conforms to the Asynchronous Module Definition (AMD) specification. For more information, see define object and SuiteScript 2.0 – Script Architecture .
Returns	Void
Supported Script Types	All script types
Governance	None
Global object	define object
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
dependencies	string Array	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 to Custom Modules.</p> <p>Note: Do not include the file extension for custom modules in the define() Function.</p>	Version 2015 Release 2
callback	Function	Required	<p>Callback function to execute after dependent modules are loaded.</p> <p>Must return entry points for SuiteScript 2.0 script types.</p>	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 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('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 {
            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
            }
        });
    }
);

```

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

```

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

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

}
);

```

define(moduleobject)

Method Description	<p>Function used to define a SuiteScript script type or define a custom module if no dependencies are required.</p> <p>If you use this function to define a SuiteScript 2.0 script type, the return statement must return at least one entry point and entry point function.</p> <p>If you use this function signature, the file name without the extension is the module name.</p> <p>For example, if you include this function in a file named <code>foo.js</code> in the SuiteScripts File Cabinet folder, use the following dependency to require it as a dependency in other script files:</p> <pre>'/SuiteScripts/foo', function(foo) ...</pre> <p>For more information, see the help topic <i>Module Dependency Paths</i>.</p>
Returns	Object
Supported Script Types	All script types
Governance	None
Global object	<code>define object</code>
Since	Version 2015 Release 2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
<code>moduleobject</code>	Object	Required	Any Javascript object.	Version 2015 Release 2

Syntax

The following code snippets show the 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 object

The require object 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, executes the callback function and loads the dependencies when they are needed.

For example, 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](#).

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. Use the require() Function to increase script performance.

Note: Only use the require() Function if you want to load a native SuiteScript 2.0 module or create or load a custom SuiteScript 2.0 module. If you want to create a SuiteScript script type, use the define() Function. See [define object](#).

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. Note: This function conforms to the Asynchronous Module Definition (AMD) specification. For more information, see require object and SuiteScript 2.0 – Script Architecture .
Returns	Void
Supported Script Types	All script types
Governance	None
Global object	require object
Since	Version 2015 Release 2

Parameters

Parameter	Type	Required / Optional	Description	Since
dependencies	string Array	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 to Custom Modules.</p> <p>Note: Do not include the file extension for custom modules in the require() Function.</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;
        }
    });
});
```

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)`
- `util.isobject(obj)`
- `util.isRegExp(obj)`
- `util.isString(obj)`

The util object also includes the following utility methods:

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

For more details about the util Object and its methods, see [N/util Module](#).

toString()

Method Description	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. 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. As a result, if you attempt to call <code>instanceOf</code> , an undefined error is thrown. 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 .
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	<p>Converts a JavaScript object value to a key-value pair string in the JSON format.</p> <p>For example, you pass the following object to JSON.stringify():</p> <pre>var contact = new object(); contact.firstName = "John"; contact.lastName = "Doe"; contact.jobTitle = "CEO";</pre> <p>This method converts it to the following string:</p> <pre>{"firstName":"John","lastName":"Doe","jobTitle":"CEO"}</pre> <p>For more information about JSON object format, see http://www.ietf.org/rfc/rfc4627.txt.</p>
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 = new object();
contact.firstName = "John";
contact.lastName = "Doe";
contact.jobTitle = "CEO";

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

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 the help topic *SuiteScript 2.0 Promise APIs*). In addition, you can create custom Promises in all client scripts (see the help topic *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
    };
});

```

Chapter 4 SuiteScript 2.0 Modules

SuiteScript 2.0 APIs are organized into the following modules:

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

Module	Description
	<ul style="list-style-type: none">• 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/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 .
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.

- [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	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.password	string	required	<ul style="list-style-type: none"> The logged in user's NetSuite password. 	Version 2015 Release 2
options.newEmail	string	required	<ul style="list-style-type: none"> The logged in user's NetSuite email address. 	Version 2015 Release 2
options.onlyThisAccount	boolean true false	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
INVALID_PSWD		The argument for <code>options.password</code> is invalid.
INVALID_EMAIL		The argument for <code>options.newEmail</code> is invalid.

Syntax

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](#).

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

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	<ul style="list-style-type: none"> The logged in user's current NetSuite password. 	Version 2015 Release 2
options.newPassword	string	required	<ul style="list-style-type: none"> 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.
USER_ERROR		

Syntax

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](#).

```
...
auth.changePassword({
    currentPassword: 'mycurrentPWD',
    newPassword: 'mynewPWD'
});
...
```

N/config Module

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

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

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

N/config Module Members

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 set the value of the Config.type property.

N/config Module Script Sample

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

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/config'],
    function(config) {
        function setTaxAndEmployerId() {
            var companyInfo = config.load({
                type: config.Type.COMPANY_INFORMATION
            });
            companyInfo.setValue({
                name: 'taxid',
                value: '1122334455'
            });
            companyInfo.setValue({
                name: 'employerid',
                value: '123456789'
            });
            companyInfo.save();
            companyInfo = config.load({
                type: config.Type.COMPANY_INFORMATION
            });
            var taxid = companyInfo.getValue({
                name: 'taxid'
            });
        }
    }
)
```

```
    setTaxAndEmployerId();
});
```

config.load(options)

Method Description	Method used to load a record.Record object that encapsulates the specified NetSuite configuration page.
Returns	record.Record
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
options.type	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
INVALID_RCRD_TYPE		The type argument is invalid or missing.

Syntax

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](#).

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

```
});  
configRecObj.save();  
...
```

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. 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/config Module
Since	Version 2015 Release 2

Values

Value	Configuration Page
USER_PREFERENCES	Set Preferences page (Home > Set Preferences)
COMPANY_INFORMATION	Company Information page (Setup > Company > Company Information > Company Information)
COMPANY_PREFERENCES	General Preferences page (Setup > Company > General Preferences)
ACCOUNTING_PREFERENCES	Accounting Preferences page (Setup > Accounting > Accounting Preferences)
ACCOUNTING_PERIODS	Accounting Periods page (Setup > Accounting > Manage Accounting Periods)
TAX_PERIODS	Tax Periods page (Setup > Accounting > Manage Tax Periods)
FEATURES	Enable Features page (Setup > Company > Enable Features)

Syntax

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](#).

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

N/crypto Module

The N/crypto module encapsulates hashing, hash-based message authentication (hmac), and symmetrical encryption.

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.
Method	crypto.createCipher(options)	object	Creates and returns a new <code>crypto.Cipher</code> object.
	crypto.createDecipher(options)	object	Creates and returns a new <code>crypto.Decipher</code> object.
	crypto.createHash(options)	object	Creates and returns a new <code>crypto.Hash</code> object.
	crypto.createHmac(options)	object	Creates and returns a new <code>crypto.Hmac</code> object.
	crypto.createSecretKey(options)	object	Creates and returns a new <code>crypto.SecretKey</code> object.
Enum	crypto.EncryptionAlg	string (read-only)	
	crypto.HashAlg	string (read-only)	
	crypto.Padding	string (read-only)	

Cipher Object Members

Member Type	Name	Return Type / Value Type	Description
Method	Cipher.update(options)	object	
	Cipher.final(options)	object	Sets the output encoding for the <code>crypto.CipherPayload</code> object.

CipherPayload Object Members

Member Type	Name	Return Type / Value Type	Description
Property	CipherPayload.encoding	string	
	CipherPayload.iv	number	An initialization vector

Decipher Object Members

Member Type	Name	Return Type / Value Type	Description
Method	Decipher.final(options)	string	
	Decipher.update(options)	void	

Hash Object Members

Member Type	Name	Return Type / Value Type	Description
Method	Hash.digest(options)	string	
	Hash.update(options)	void	

Hmac Object Members

Member Type	Name	Return Type / Value Type	Description
Method	Hmac.digest(options)	string	
	Hash.update(options)	void	

SecretKey Object Members

Member Type	Name	Return Type / Value Type	Description
Property	Secretkey.guid		
	SecretKey.encoding		

N/crypto Module Script Samples

The following examples generate a secure key using the SHA512 hashing algorithm. Note that the first example is meant to show how to use the APIs but will not actually work in the debugger because the GUID does not exist in your account. Please try the Suitelet example for a more complete usage.

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/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();
});
```

Note: This sample script 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', 'N/crypto', 'N/encode'],
    function(ui, crypto, encode) {
        function onRequest(option) {
            if (option.request.method === 'GET') {
                var form = ui.createForm({
                    title: 'My credential form'
                });
                form.addSecretKeyField({
                    id: 'mycredential',
                    label: 'Credential',
                    restrictToScriptId: 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(e.name);
    }
    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.
Supported Script Types	Server-side scripts
Module	N/crypto 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/crypto Module Script Samples](#).

Cipher.final(options)

Method Description	
Returns	A crypto.CipherPayload 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
options.outputEncoding	string	required	The output encoding for a crypto.CipherPayload object.

Syntax

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](#).

Cipher.update(options)

Method Description	
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 cipher data to be updated.
options.inputEncoding	string	optional	<p>The input encoding. Use the encode.Encoding enum to set the value. The default value is <code>UTF_8</code>.</p>

Syntax

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](#).

crypto.CipherPayload

Object Description	Encapsulates a cipher payload.
Supported Script Types	Server-side scripts
Module	N/crypto 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/crypto Module Script Samples](#).

CipherPayload.encoding

Property Description	
Type	string
Module	N/crypto 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/crypto Module Script Samples](#).

CipherPayload.iv

Property Description	
Type	string
Module	N/crypto 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/crypto Module Script Samples](#).

crypto.Decipher

Object Description	
Supported Script Types	Server-side scripts
Module	N/crypto 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/crypto Module Script Samples](#).

Decipher.final(options)

Method Description	
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	required	

Syntax

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](#).

Decipher.update(options)

Method Description	Method used to update decipher 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		
options.inputEncoding	string		

Syntax

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](#).

crypto.Hash

Object Description	
Supported Script Types	Server-side scripts
Module	N/crypto 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/crypto Module Script Samples](#).

Hash.digest(options)

Method Description	Calculates the digest of the data to be hashed.
Returns	A buffer or 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 which can be set by the encode.Encoding enum. The default value is <code>UTF_8</code> .

Syntax

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](#).

Hash.update(options)

Method Description	Method used to update hash 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 hash data to be updated.

Parameter	Type	Required / Optional	Description
options.inputEncoding	string	optional	The input encoding which can be set by the encode.Encoding enum. The default value is <code>UTF_8</code> .

Syntax

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](#).

crypto.Hmac

Object Description	Encapsulates an hmac.
Supported Script Types	Server-side scripts
Module	N/crypto 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/crypto Module Script Samples](#).

Hmac.digest(options)

Method Description	Gets the computed digest.
Returns	A buffer or 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		

Syntax

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](#).

Hmac.update(options)

Method Description	Method used to update the hmac 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	string	optional	The input encoding which can be set by the encode.Encoding enum. The default value is <code>UTF_8</code> .

Syntax

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](#).

crypto.SecretKey

Object Description	
Supported Script Types	Server-side scripts
Module	N/crypto 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/crypto Module Script Samples](#).

SecretKey.encoding

Property Description	
Type	string
Module	N/crypto 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/crypto Module Script Samples](#).

Secretkey.guid

Property Description	
Type	string
Module	N/crypto 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/crypto Module Script Samples](#).

crypto.createCipher(options)

Method Description	Method used to create and return a new crypto.EncryptionAlg object.
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.blockCipherMode	string	required	The encryption mode for the cipher block	Version 2015 Release 2
options.padding	string	required	The padding for the cipher. Set the value using the crypto.Padding enum.	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/crypto Module Script Samples](#).

crypto.createDecipher(options)

Method Description	Method used to create a new <code>crypto.Decipher</code> object.
Returns	A <code>crypto.Decipher</code> 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
<code>options.algorithm</code>	string	required	The hash algorithm. Set by the <code>crypto.Hash</code> enum.	Version 2015 Release 2
<code>options.key</code>	object	required	The <code>crypto.SecretKey</code> object.	Version 2015 Release 2
<code>options.blockCipherMode</code>	object	required		Version 2015 Release 2
<code>options.padding</code>	object	required		Version 2015 Release 2
<code>options.iv</code>	string	required		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/crypto Module Script Samples](#).

crypto.createHash(options)

Method Description	Method used to create a new <code>crypto.Hash</code> object.
Returns	The <code>crypto.Hash</code> 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	<ul style="list-style-type: none"> The hash algorithm. Set by the crypto.Hash enum. 	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/crypto Module Script Samples](#).

crypto.createHmac(options)

Method Description	Method used to create a new 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. Set by the crypto.Hash enum.	Version 2015 Release 2
options.key	object	required	The crypto.SecretKey object.	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/crypto Module Script Samples](#).

crypto.createSecretKey(options)

Method Description	Method used to create a new crypto.SecretKey object.
Returns	An crypto.SecretKey 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.guid	string	required		Version 2015 Release 2
options.encoding	string	optional		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/crypto Module Script Samples](#).

crypto.EncryptionAlg

Enum Description	
Module	N/crypto Module
Since	Version 2015 Release 2

Values

- AES

Syntax

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](#).

crypto.HashAlg

Enum Description	Sets the value of the <code>options.algorithm</code> parameter for <code>crypto.createHash(options)</code> . Note: Enum values are constants and therefore read-only.
Module	N/crypto Module
Since	Version 2015 Release 2

Values

- SHA1

- SHA256
- SHA512
- MD5

Syntax

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](#).

crypto.Padding

Enum Description	
Module	N/crypto Module
Since	Version 2015 Release 2

Values

- NoPadding
- PKCS5Padding

Syntax

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](#).

N/currency Module

Load the N/currency module when you want to work with exchange rates within your NetSuite account. You can use this module to find the exchange rate between two currencies based on a certain date.

To use multiple currencies, the Multiple Currencies feature must be enabled. For information on enabling this feature, see the help topic [Enabling the Multiple Currencies Feature](#).

Note: Currency formatting is handled by the [N/format Module](#).

- [N/currency Module Member](#)
- [N/currency Module Script Sample](#)

N/currency Module Member

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.

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. 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 date is not specified, then it defaults to today (the current date). The date determines the exchange rate in effect. If there are multiple rates, it is the latest entry on that date. Use the same date format as your NetSuite account. 	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.
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

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](#).

...

```

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

```

N/email Module

Load the N/email module when you want to send email messages from within NetSuite. You can use the N/email module to send regular, bulk, and campaign email.

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

N/email Module Members

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.

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.
Returns	void
Supported Script Types	Client and server-side scripts
Governance	20 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
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. A maximum of 10 recipients (recipient + cc + bcc) is allowed. <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>	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	string[]	optional	<ul style="list-style-type: none"> Internal IDs or an array of email addresses of the secondary recipients to copy. A maximum of 10 recipients (recipient + cc + bcc) is allowed. 	Version 2015 Release 2
options.bcc	string[]	optional	<ul style="list-style-type: none"> Internal IDs, email entity, or an array of email addresses to blind copy 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

Parameter	Type	Required / Optional	Description	Since
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. <p>To display XML as plain text, use an HTML<pre> tag around XML content.</p>	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 5MB 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
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) attached to the Message record	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			Use for transaction and opportunity record types.	
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 Note: If you use this parameter, customRecord.recordType is required.	Version 2015 Release 2
customRecord.recordType	string	optional	The custom record type attached to the Message record Note: If you use this parameter, customRecord.id is required.	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/email Module Script Sample](#).

```
...
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: '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
    }
});
...

```

email.send.promise(options)

Method Description	Method used to send transactional email asynchronously and receive bounceback notifications if the email is not successfully delivered.
	Note: For information about the parameters and errors thrown for this method, see email.send(options) . For additional information about promises, see Promise object .
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.
	Note: This API normally uses a bulk email server to send messages. If you need to increase the successful delivery rate of an email, use email.send(options) so that a transactional email server is used.
Returns	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
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 email addresses. <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>	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	string[]	optional	<ul style="list-style-type: none"> Internal IDs or an array of email addresses of the secondary recipients to copy. 	Version 2015 Release 2
options.bcc	string[]	optional	<ul style="list-style-type: none"> Internal IDs, email entity, or an array of email addresses to blind copy 	Version 2015 Release 2
options.subject	string	required	<ul style="list-style-type: none"> Subject of the outgoing message. 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
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. <p>To display XML as plain text, use an HTML<pre> tag around XML content.</p>	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 5MB 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
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 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> Use for transaction and opportunity record types. 	
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

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](#).

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

email.sendBulk.promise(options)

Method Description	This method is used to send bulk email asynchronously when a bounceback notification is not required.
	<p>Note: For information about the parameters and errors thrown for this method, see email.sendBulk(options). For additional information about promises, see Promise object.</p>
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 email.send(options) so that a transactional email server is used.
Returns	A campaign response ID (tracking code) as number If the email fails to send, the value returned is –1.
Supported Script Types	Client and server-side scripts
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.campaignEventId	number	required	• The internal ID of the campaign event.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			Note: The campaign must use a Lead Nurturing (campaignndrip) sublist.	
options.recipientId	number	required	<ul style="list-style-type: none"> The internal ID of the recipient. Note: The recipient's record must contain an email address.	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/email Module Script Sample](#).

```
...
email.sendCampaignEvent({
    campaignEventId: -8,
    recipientId: 142,
});
...
```

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. Note: For information about the parameters and errors thrown for this method, see email.sendCampaignEvent(options) . For additional information about promises, see Promise object .
Returns	A campaign response ID (tracking code) as a number. If the email fails to send, the value returned is -1.
Synchronous Version	<code>email.sendCampaignEvent(options)</code>
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.

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 = "Tést Stríng 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 UTF_8. Use the encode.Encoding to set the value.	Version 2015 Release 1
options.outputEncoding	string	required	The encoding to apply to the output string. The default value is UTF_8. Use the encode.Encoding to set the value.	Version 2015 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/encode Module Script Sample](#).

```
...
var hexEncodedString = encode.convert({
    string: stringInput,
    inputEncoding: encode.Encoding.UTF_8,
    outputEncoding: encode.Encoding.HEX
});
...
```

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

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](#).

```
...
var reencoded = encode.convert({
    string: LOREM_IPS,
    inputEncoding: encode.Encoding.BASE_64,
    outputEncoding: encode.Encoding.UTF_8
});
...
```

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/file 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.

Member Type	Name	Return Type / Value Type	Description
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.

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. Note: When <code>file.create(options)</code> is called in a user event script, it returns <code>error.UserEventError</code> .
Supported Script Types	All server-side scripts that are not user event scripts.
Module	N/error 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/error Module Script Sample](#).

```
...  
var errorObj = error.create({  
    name: 'MY_CODE',  
    message: 'my error details',  
    notifyOff: false  
});  
...
```

SuiteScriptError.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
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/error Module Script Sample](#).

```
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error ID: " + errorObj.id);
...
```

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

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](#).

```
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Message: " + errorObj.message);
...
```

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

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](#).

```
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Code: " + errorObj.name);
...
```

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

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](#).

```
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Stack: " + errorObj.stack);
...
```

error.UserEventError

Object Description	Encapsulates a SuiteScript error for user event scripts. Use this object in a try-catch statement to abort script execution.
---------------------------	---

	<p>Create a new custom error (<code>error.UserEventError</code>) with the <code>file.create(options)</code> method.</p> <p>The <code>file.create(options)</code> method returns <code>error.UserEventError</code> when it is called in a user event script.</p> <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

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](#).

```
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
```

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

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](#).

```
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
```

```
log.debug("User Event Type: " + errorObj.eventType);
...
```

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

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](#).

```
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Stack: " + errorObj.stack);
...
```

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
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/error Module Script Sample](#).

```
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error ID: " + errorObj.id);
...
```

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

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](#).

```
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Message: " + errorObj.message);
...
```

UserEventError.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

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](#).

```
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Code: " + errorObj.name);
...
```

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

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](#).

```
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Submitted Record ID: " + errorObj.recordId);
...
```

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

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	• A user-defined name (error code).	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> Sets the value for the property <code>file.load(options)</code>. 	
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>. 	Version 2015 Release 2
options.notifyOff	boolean <code>true</code> <code>false</code>	optional	<ul style="list-style-type: none"> Sets whether email notification is suppressed. The default value is false. If set to false, when this error is thrown, the system emails the users identified on the applicable script record's Unhandled Errors tab. 	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
MISS_MANDATORY_PARAMETER		A required argument is missing

Syntax

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](#).

```
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Code: " + errorObj.name);
...
```

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.

- [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	Used to set the value of the File.encoding property.
	file.Type	enum	Used to set 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.getContents()	string	Returns the content of a file in string format.
	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.
	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.

Member Type	Name	Return Type / Value Type	Description
	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

The following example creates and saves a file to the file cabinet.

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/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();
    });

```

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 <code>File.getContents()</code> method. Important: Binary content must be base64 encoded Create a new <code>file.File</code> (up to 10MB in size) with the <code>file.create(options)</code> method.
---------------------------	---

	<p>After you create a new <code>file.File</code>, you can:</p> <ul style="list-style-type: none"> upload it to the NetSuite file cabinet with the <code>File.save()</code> method. attach it to an email or fax without saving it to the file cabinet. <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>
Supported Script Types	Server-side scripts
Module	N/file 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/file Module Script Sample](#).

```
...
var fileObj = file.create({
    name   : 'test.txt',
    fileType: file.Type.PLAINTEXT,
    contents: 'Hello World\nHello World'
});
fileObj.folder = 30;
var fileId = fileObj.save();
...

```

File.getContents()

Method Description	Method used to return the content of the file.
	Important: Content is lazy loaded and must be less than 10MB in size to access with this method.
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

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](#).

```
...
var fileObj = file.load({idOrPath: 145});
if (fileObj.size < 10485760){
    fileObj.getContents();
}
...
```

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 can upload or save files of any size, as long as the file size is permitted by the file cabinet.</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

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](#).

```
...
```

```
var fileObj = file.create({
    name   : 'test.text',
    fileType: file.Type.PLAINTEXT,
    contents: 'Hello World\nHello World'
});
fileObj.folder = 30;
var fileId = fileObj.save();
...
```

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

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](#).

```
...
var fileObj = file.load({idOrPath: '/Images/myImageFile.jpg'});
fileObj.description = 'my test file';
var fileId = fileObj.save();
...
```

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

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](#).

```
...
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();
...
```

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

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](#).

```
...
var fileObj = file.load({idOrPath: 145});
log.debug("File Type: " + fileObj.fileType);
...
```

File.folder

Property Description	The internal ID of a file's folder within the NetSuite file cabinet. Before you upload a file to the NetSuite file cabinet with File.save() , you must set its file cabinet folder with the folder property.
Type	number string
Module	N/file 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/file Module Script Sample](#).

```
...
var fileObj = file.create({
    name      : 'test.text',
    fileType: file.Type.PLAINTEXT,
    contents: 'Hello World\nHello World'
});
fileObj.folder = 30;
var fileId = fileObj.save();
...

```

File.id

Property Description	The internal ID of the file within the NetSuite file cabinet. This value is automatically generated by NetSuite. This property is read-only.
Type	number
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

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](#).

```
...
var fileObj = file.load({idOrPath: '/Images/myImageFile.jpg'});
log.debug("File ID: " + fileObj.id);
...
```

File.isInactive

Property Description	The inactive status of a file. If set to true, the file is inactive. The default value is false. When a file is inactive, it does not display in the UI unless you select Show Inactives on the File Cabinet page.
Type	boolean true false
Module	N/file 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/file Module Script Sample](#).

```
...
var fileObj = file.load({idOrPath: '/Images/myImageFile.jpg'});
fileObj.name = 'myOldImageFile.jpg';
fileObj.isInactive = true;
var fileId = fileObj.save();
...
```

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> . 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.
Type	<code>boolean true false</code>
Module	N/file 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/file Module Script Sample](#).

```
...
var fileObj = file.load({idOrPath: '/Images/myImageFile.jpg'});
fileObj.isOnLine = true;
var fileId = fileObj.save();
...
```

File.isText

Property Description	Indicates whether a file type is text-based. This property is read-only.
Type	<code>boolean true false</code>
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

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](#).

```
...
var fileObj = file.load({idOrPath: 145});
if (fileObj.isText === true){
    ...
}
...
```

File.name

Property Description	The name of a file.
Type	string
Module	N/file 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/file Module Script Sample](#).

```
...
var fileObj = file.load({idOrPath: '/Images/myImageFile.jpg'});
fileObj.name = 'myOldImageFile.jpg';
var fileId = fileObj.save();
...
```

File.path

Property Description	The relative path to a file in the NetSuite file cabinet. 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

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](#).

```
...
var fileObj = file.load({idOrPath: 145});
log.debug("File Path: " + fileObj.path);
...
```

File.size

Property Description	The size of a file in bytes. 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

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](#).

```
...
var fileObj = file.load({idOrPath: '/Images/myImageFile.jpg'});
log.debug("File Size: " + fileObj.size);
...
```

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

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](#).

```
...
var fileObj = file.load({idOrPath: '/Images/myImageFile.jpg'});
log.debug("File URL: " + fileObj.url);
...
```

file.create(options)

Method Description	Method used to create a new file in the NetSuite file cabinet.
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. 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	required	<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

Parameter	Type	Required / Optional	Description	Since
options.folder	number	optional	The internal ID of the folder used when the file is saved	Version 2016 Release 1

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.

Syntax

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](#).

```
...
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(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

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. 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> To find the internal ID of the file in the UI, click Documents > Files > File Cabinet. 	

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	<name of missing parameter>	A required argument is not passed.

Syntax

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](#).

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

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.idOrPath	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. 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> 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. 	

Errors

Error Code	Message	Thrown If
INSUFFICIENT_PERMISSION	You do not have access to the media item you selected.	Internal ID passed is invalid.
RCRD_DSNT_EXIST	That record does not exist. path: {path}	Relative file path passed is invalid.
SSS_MISSING_REQD_ARGUMENT	<name of missing parameter>	A required argument is not passed.

Syntax

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](#).

```
...
var fileObj = file.load({idOrPath: '/Images/myImageFile.jpg'});
fileObj.description = 'my test file';
var fileId = fileObj.save();
...
```

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. 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/file Module
Since	Version 2015 Release 2

Values

Value	Character Set
UTF8	Unicode

Value	Character Set
WINDOWS_1252	Western
ISO_8859_1	Western
GB18030	Chinese Simplified
SHIFT_JIS	Japanese
MAC_ROMAN	Western
GB2312	Chinese Simplified
BIG5	Chinese Traditional

Syntax

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](#).

```
...
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();
...
```

file.Type

Enum Description	Enumeration that holds the string values for supported file types. This enum is used to set the value of the File.fileType property. Note that the <code>File.fileType</code> property is read only. Its value must be set with <code>file.create(options)</code> . See N/file Module Script Sample for an example. Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Module	N/file Module
Since	Version 2015 Release 2

Values

- | | | |
|--|---|--|
| <ul style="list-style-type: none"> • AUTOCAD • BMPIMAGE • CSV | <ul style="list-style-type: none"> • JSON • MESSAGERFC • MP3 | <ul style="list-style-type: none"> • RTF • SMS • STYLESHEET |
|--|---|--|

• EXCEL	• MPEGMOVIE	• TAR
• FLASH	• MSPROJECT	• TIFFIMAGE
• FREEMARKER	• PDF	• VISIO
• GIFIMAGE	• PJPGIMAGE	• WEBAPPPAGE
• GZIP	• PLAINTEXT	• WEBAPPSCRIPT
• HTMLDOC	• PNGIMAGE	• WORD
• ICON	• POSTSCRIPT	• XMLDOC
• JAVASCRIPT	• POWERPOINT	• XSD
• JPGIMAGE	• QUICKTIME	• ZIP

Syntax

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](#).

```
...
var fileObj = file.create({
    name : 'test.txt',
    fileType: file.Type.PLAINTEXT,
    contents: 'Hello World\nHello World'
});
fileObj.folder = 30;
var fileId = fileObj.save();
...

```

N/format Module

You can use the format module to parse formatted data into strings and to convert strings into a specified format.

- [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. Note: This method is overloaded when you format a <code>datetime</code> or <code>datetimetz</code> value.
	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 <code>datetime</code> or <code>datetimetz</code> value.
Enum	format.Type	enum	Holds the string values for the supported field types. This enum is used to set the value of the <code>options.type</code> parameter.
	format.Timezone	enum	Holds the string values for supported time zone formats. This enum is used to set the value of the <code>options.timezone</code> parameter.

N/format Module Script 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	Method used to format a value from the raw value to its appropriate preference format
Description	

	Note: This method is overloaded when you format a <code>datetime</code> or <code>datetimetz</code> value.
Returns	<p>The formatted value as a string.</p> <p>If a <code>datetime</code> or <code>datetimetz</code> value was given, the Date Object is returned in the user's local app time zone. Returns NaN if the date includes a leading zero.</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>	<code>Date</code> <code>string</code> <code>number</code>	required	The input data to format.	Version 2015 Release 2
<code>options.type</code>	<code>string</code>	required	<p>The field type (for example, <code>DATE</code>, <code>CURRENCY</code>, <code>INTEGER</code>).</p> <p>Set using the format.Type enum.</p>	Version 2015 Release 2

The table below applies to `datetime` and `datetimetz` values only.

Parameter	Type	Required / Optional	Description	Since
<code>options.value</code>	<code>Date</code>	required	The Date Object being converted into a string	Version 2015 Release 2
<code>options.type</code>	<code>string</code>	required	<p>The field type (either <code>DATETIME</code> or <code>DATETIMETZ</code>).</p> <p>Set using the format.Type enum.</p>	Version 2015 Release 2
<code>options.timezone</code>	<code>enum</code> <code>number</code>	optional	<p>The time zone specified for the returned 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>	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			If the time zone is invalid, the time zone is set to GMT.	

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](#).

```
...
var formattedDateString = format.format({
    value: parsedDateStringAsRawDateObject,
    type: format.Type.DATE
});
...
...
```

format.parse(options)

Method Description	Method used to parse a value from the appropriate preference format to its raw value. 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. Note: This method is overloaded when you format a <code>datetime</code> or <code>datetimetz</code> value.
Returns	The parsed value as a <code>Date</code> <code>string</code> <code>number</code> <code>Datetime</code> or <code>datetimetz</code> values are returned as a string. Returns <code>Nan</code> if the date includes a leading zero. Note: If the value given is not valid or parsable, the original value passed to <code>options.value</code> is returned.
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>	<code>string</code>	required	The input data to parse.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.type	string	required	The field type (for example, DATE, CURRENCY, INTEGER). Set using the format.Type enum.	Version 2015 Release 2

The table below applies to datetime and datetimeTZ values only.

Parameter	Type	Required / Optional	Description	Since
options.value	string	required	The string that contains the date and time information in the specified timezone. Note: Leading zeroes in the month and day values are not supported.	Version 2015 Release 2
options.type	string	required	The field type (either DATETIME or DATETIMETX). Set using the format.Type enum.	Version 2015 Release 2
options.timezone	enum	optional	The time zone represented by the options.value string. Set using the format.Timezone 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

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](#).

```
...
var initialFormattedDateString = "07/28/2015";
var parsedDateStringAsRawDateObject = format.parse({
    value: initialFormattedDateString,
    type: format.Type.DATE
});
...
```

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> .
-------------------------	--

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/format Module
Since	Version 2015 Release 2

Values

<ul style="list-style-type: none"> • CCEXDATE • CCNUMBER • CCVALIDFROM • CHECKBOX • COLOR • CURRENCY • CURRENCY2 • DATE • DATETIME • DATETIMETZ 	<ul style="list-style-type: none"> • FLOAT • FULLPHONE • FUNCTION • IDENTIFIER • INTEGER • MMYYDATE • NONNEGURRENCY • NONNEGFLOAT • PERCENT • PHONE 	<ul style="list-style-type: none"> • POSCURRENCY • POSFLOAT • POSINTEGER • RATE • RATEHIGHPRECISION • TIME • TIMEOFDAY • TIMETRACK • URL
---	---	---

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 options.timezone parameter when calling <code>format.format(options)</code> or <code>format.parse(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/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
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

Key	Olson Value	Description
32	GMT: 'GMT'	(GMT) Casablanca
33	ATLANTIC_REYKJAVIK: 'Atlantic/Reykjavik'	(GMT) Monrovia, Reykjavik
34	EUROPE_WARSAW: 'Europe/Warsaw'	(GMT+01:00) Sarajevo, Skopje, Warsaw, Zagreb
35	EUROPE_PARIS: 'Europe/Paris'	(GMT+01:00) Brussels, Copenhagen, Madrid, Paris
36	ETC_GMT_MINUS_1: 'Etc/GMT-1'	(GMT+01:00) West Central Africa
37	EUROPE_AMSTERDAM: 'Europe/Amsterdam'	(GMT+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
38	EUROPE_BUDAPEST: 'Europe/Budapest'	(GMT+01:00) Belgrade, Bratislava, Budapest, Ljubljana, Prague
39	AFRICA_CAIRO: 'Africa/Cairo'	(GMT+02:00) Cairo
40	EUROPE_ISTANBUL: 'Europe/Istanbul'	(GMT+02:00) Athens, Bucharest, Istanbul
41	ASIA_JERUSALEM: 'Asia/Jerusalem'	(GMT+02:00) Jerusalem
42	ASIA_AMMAN: 'Asia/Amman'	(GMT+02:00) Amman
43	ASIA_BEIRUT: 'Asia/Beirut'	(GMT+02:00) Beirut
44	AFRICA_JOHANNESBURG: 'Africa/Johannesburg'	(GMT+02:00) Harare, Pretoria
45	EUROPE_KIEV: 'Europe/Kiev'	(GMT+02:00) Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius
46	EUROPE_MINSK: 'Europe/Minsk'	(GMT+02:00) Minsk
47	AFRICA_WINDHOEK: 'Africa/Windhoek'	(GMT+02:00) Windhoek
48	ASIA_RIYADH: 'Asia/Riyadh'	(GMT+03:00) Kuwait, Riyadh
49	EUROPE_MOSCOW: 'Europe/Moscow'	(GMT+03:00) Moscow, St. Petersburg, Volgograd
50	ASIA_BAGHDAD: 'Asia/Baghdad'	(GMT+03:00) Baghdad
51	AFRICA_NAIROBI: 'Africa/Nairobi'	(GMT+03:00) Nairobi
52	ASIA_TEHRAN: 'Asia/Tehran'	(GMT+03:30) Tehran
53	ASIA_MUSCAT: 'Asia/Muscat'	(GMT+04:00) Abu Dhabi, Muscat
54	ASIA_BAKU: 'Asia/Baku'	(GMT+04:00) Baku
55	ASIA_YEREVAN: 'Asia/Yerevan'	(GMT+04:00) Caucasus Standard Time
56	ETC_GMT_MINUS_3: 'Etc/GMT-3'	(GMT+04:00) Tbilisi
57	ASIA_KABUL: 'Asia/Kabul'	(GMT+04:30) Kabul
58	ASIA_KARACHI: 'Asia/Karachi'	(GMT+05:00) Islamabad, Karachi
59	ASIA_YEKATERINBURG: 'Asia/Yekaterinburg'	(GMT+05:00) Ekaterinburg
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

Key	Olson Value	Description
63	ASIA_ALMATY: 'Asia/Almaty'	(GMT+06:00) Novosibirsk
64	ASIA_DHAKA: 'Asia/Dhaka'	(GMT+06:00) Astana, Dhaka
65	ASIA_RANGOON: 'Asia/Rangoon'	(GMT+06:30) Yangon (Rangoon)
66	ASIA_BANGKOK: 'Asia/Bangkok'	(GMT+07:00) Bangkok, Hanoi, Jakarta
67	ASIA_KRASNOYARSK: 'Asia/Krasnoyarsk'	(GMT+07:00) Krasnoyarsk
68	ASIA_HONG_KONG: 'Asia/Hong_Kong'	(GMT+08:00) Beijing, Chongqing, Hong Kong, Urumqi
69	ASIA_KUALA_LUMPUR: 'Asia/Kuala_Lumpur'	(GMT+08:00) Kuala Lumpur, Singapore
70	ASIA_TAIPEI: 'Asia/Taipei'	(GMT+08:00) Taipei
71	AUSTRALIA_PERTH: 'Australia/Perth'	(GMT+08:00) Perth
72	ASIA_IRKUTSK: 'Asia/Irkutsk'	(GMT+08:00) Irkutsk
73	ASIA_MANILA: 'Asia/Manila'	(GMT+08:00) Manila
74	ASIA_SEOUL: 'Asia/Seoul'	(GMT+09:00) Seoul
75	ASIA_TOKYO: 'Asia/Tokyo'	(GMT+09:00) Osaka, Sapporo, Tokyo
76	ASIA_YAKUTSK: 'Asia/Yakutsk'	(GMT+09:00) Yakutsk
77	AUSTRALIA_DARWIN: 'Australia/Darwin'	(GMT+09:30) Darwin
78	AUSTRALIA_ADELAIDE: 'Australia/Adelaide'	(GMT+09:30) Adelaide
79	AUSTRALIA_SYDNEY: 'Australia/Sydney'	(GMT+10:00) Canberra, Melbourne, Sydney
80	AUSTRALIA_BRISBANE: 'Australia/Brisbane'	(GMT+10:00) Brisbane
81	AUSTRALIA_HOBART: 'Australia/Hobart'	(GMT+10:00) Hobart
82	PACIFIC_GUAM: 'Pacific/Guam'	(GMT+10:00) Guam, Port Moresby
83	ASIA_VLADIVOSTOK: 'Asia/Vladivostok'	(GMT+10:00) Vladivostok
84	ASIA_MAGADAN: 'Asia/Magadan'	(GMT+11:00) Magadan, Solomon Is., New Caledonia
85	PACIFIC_KWAJALEIN: 'Pacific/Kwajalein'	(GMT+12:00) Fiji, Marshall Is.
86	PACIFIC_AUCKLAND: 'Pacific/Auckland'	(GMT+12:00) Auckland, Wellington
87	PACIFIC_TONGATAPU: 'Pacific/Tongatapu'	(GMT+13:00) Nuku'alofa

Syntax

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](#).

N/http Module

Use the http module to make http calls from server-side or client-side scripts.

The http module does not accept the HTTPS protocol.

Note: All HTTP content types are supported.

- N/http Module Members
- ClientResponse Object Members
- ServerRequest Object Members
- ServerResponse Object Members
- N/http Module Script Sample

N/http Module Members

Member Type	Name	Return Type / Value Type	Description
Object	http.ClientResponse	read-only Object	Encapsulates the result of an http request from an HTTP client.
	http.ServerRequest	read-only Object	Encapsulates the incoming http request information for an HTTP server.
	http.ServerResponse	Object	Encapsulates the response to an incoming http request for an HTTP server.
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.

Member Type	Name	Return Type / Value Type	Description
	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 ServerResponse.setCdnCacheable(options) property.
	http.Method	enum	Holds the string values for supported HTTP requests. This enum is used to set the value of the ClientResponse.method and ServerRequest.method properties.

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](#) 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	string	The server request body
	ServerRequest.files	Object	The server request files.
	ServerRequest.headers	Object	The server request headers.
	ServerRequest.method	string	The http method for the server request.
	ServerRequest.parameters	Object	The server request parameters.
	ServerRequest.url	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	<code>ServerResponse.addHeader(options)</code>	void	Adds a header to the response
	<code>ServerResponse.getHeader(options)</code>	void	Returns the value of a response header
	<code>ServerResponse.renderPdf(options)</code>	void	Generates and renders a PDF directly to the response
	<code>ServerResponse.sendRedirect(options)</code>	void	Sets the redirect URL by resolving to a NetSuite resource
	<code>ServerResponse.setCdnCacheable(options)</code>	void	Sets CDN caching for a period of time.
	<code>ServerResponse.setHeader(options)</code>	void	Sets the value of a response header.
	<code>ServerResponse.write(options)</code>	void	Writes information (text/xml/html) to the response.
	<code>ServerResponse.writeFile(options)</code>	void	Writes a file to the response.
	<code>ServerResponse.writeLine(options)</code>	void	Writes line information (text/xml/html) to the response.
	<code>ServerResponse.writePage(options)</code>	void	Generates a page.
Property	<code>ServerResponse.headers</code>	Object	The server response headers.

N/http Module Script Sample

The following example shows an HTTP GET request for a URL.

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

```
});
```

http.ClientResponse

Object Description	Encapsulates the result of an http request from an HTTP client. This object is read-only.
Supported Script Types	Server-side scripts
Module	N/http Module
Since	Version 2015 Release 2

ClientResponse.body

Property Description	The 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.

ClientResponse.code

Property Description	The 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.

ClientResponse.headers

Property Description	The response 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.

ClientResponse.method

Property Description	The name of the server request HTTP method. 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.

ClientResponse.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.

ClientResponse.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.

http.ServerRequest

Object Description	Encapsulates the incoming http request information for an HTTP server. This object is read-only.
Supported Script Types	Server-side scripts
Module	N/http Module
Since	Version 2015 Release 2

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.

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 name of the field.	Version 2015 Release 2
options.line	string	required	The line number.	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.

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.

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.

ServerRequest.headers

Property Description	The server request 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.

ServerRequest.method

Property Description	The server request http method. 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.

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.

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.

http.ServerResponse

Object Description	Encapsulates the response to an incoming http request for an HTTP server.
Supported Script Types	Server-side scripts
Module	N/http Module
Since	Version 2015 Release 2

ServerResponse.addHeader(options)

Method Description	Method used to add a header to the response. If the same header has already been set, this method adds another line for that header. For example:
	Vary: 'Accept-Language' Vary: 'Accept-Encoding'
Returns	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.

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.

ServerResponse.sendRedirect(options)

Method Description	Method used to set the redirect URL by resolving to a NetSuite resource.
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.

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: • RECORD	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> TASKLINK SUITELET 	
options.identifier	string	required	<p>The primary ID for this resource.</p> <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 SUITLET, input the script ID. 	Version 2015 Release 2
options.id	string	optional	<p>The secondary ID for this resource. If the base type is SUITLET, input the deployment ID.</p>	Version 2015 Release 2
options.editMode	boolean true false	optional	If the base type is RECORD, this value determines whether to return a URL for the record in edit mode or view mode.	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.	<p>The value input for options.type is not one of the following:</p> <ul style="list-style-type: none"> RECORD TASKLINK SUITELET
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.

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.

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/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.

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	string	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.

ServerResponse.write(options)

Method Description	Method used to write information (text/xml/html) 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 or file 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.

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.isInline	boolean true false	optional	Determines whether the field is inline. If true, the file is inline.	

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.

ServerResponse.writeLine(options)

Method Description	Method used to write line information (text/xml/html) 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.

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.F	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.

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: <code>{Vary: ['Accept-Language', 'Accept-Encoding']}</code>

Module	N/http Module
Since	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

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

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](#).

http.get.promise(options)

Method Description	Method used to send an HTTP GET request asynchronously and return server 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

http.delete(options)

Method Description	Method used to send an HTTP DELETE request and returns the response. Note: This method does not include an <code>options.body</code> parameter. Postdata is not required when the HTTP method is a <code>DELETE</code> request.
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
<code>options.url</code>	string	required	The HTTP URL being requested	Version 2015 Release 2
<code>options.headers</code>	object	optional	The HTTP headers.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	Missing a required argument: {param name}	A required parameter is not passed.

Syntax

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](#).

http.delete.promise(options)

Method Description	Method used to send an HTTP DELETE request asynchronously and return server response.
	<p>Note: For information about the parameters and errors thrown for this method, see http.delete(options). For additional information on promises, see Promise object.</p>
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

http.request(options)

Method Description	Method used to send an HTTP 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.method	enum	required	The HTTP request method. Set using the http.Method enum.	Version 2015 Release 2
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. <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

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](#).

http.request.promise(options)

Method Description	Method used to send an HTTP request asynchronously and return server 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

http.post(options)

Method Description	Method used to send an HTTP PUT request and return server 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

Parameter	Type	Required / Optional	Description	Since
options.body	string object	required	The POST data.	
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

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](#).

http.post.promise(options)

Method Description	Method used to send an HTTP PUT request asynchronously and return server 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

http.put(options)

Method Description	Method used to send na HTTP PUT request and return server response.
Returns	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 PUT data.	
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

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](#).

http.put.promise(options)

Method Description	Method used to send an HTTP PUT request asynchronously and return server response. 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

http.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.
-------------------------	--

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

- LONG
- MEDIUM
- SHORT
- UNIQUE

Syntax

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](#).

http.Method

Enum Description	Holds the string values for supported HTTP requests. This enum is used to set the value of the ClientResponse.method and ServerRequest.method properties.
Module	N/http Module

Values

- DELETE
- GET
- PUT
- POST

Syntax

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](#).

N/https Module

Load the https module when you need to manage content sent to a third party via https calls. You can use this module to encode binary content or access a handle to the value in a NetSuite

credential field. You can also use the https Module to perform various string transformations using methods that hash, encode, or append another string. SecureString functionality is supported only in server-side scripts.

You can make https calls from client and server-side scripts. This module encapsulates all the functionality of the [N/http Module](#), but does not allow the HTTP protocol.

Note: When the https module is used, SuiteScript also loads the [N/crypto Module](#) and [N/encode Module](#).

- [N/https Module Members](#)
- [SecureString Object Members](#)
- [ClientResponse Object Members](#)
- [ServerResponse Object Members](#)
- [ServerRequest Object Members](#)
- [N/https Module Script Sample](#)

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 result of an https request from an HTTPS client.
	https.ServerRequest	read-only Object	Encapsulates the incoming https request information for an HTTPS server.
	https.ServerResponse	Object	Encapsulates the response to an incoming https request for an HTTPS server.
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.get(options)	https.ClientResponse	Sends an HTTPS GET request and returns the response.

Member Type	Name	Return Type / Value Type	Description
	https.post(options)	https.ClientResponse	Sends an HTTPS POST request and returns the response.
	https.put(options)	https.ClientResponse	Sends an HTTPS PUT request and returns the response.
	https.request(options)	https.ClientResponse	Sends an HTTPS request 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.Method	enum	Holds the string values for supported HTTP requests. This enum is used to set the value of the ClientResponse.method and ServerRequest.method properties.

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	<code>ServerRequest.getLineCount(options)</code>	number	Returns the number of lines in a sublist.
	<code>ServerRequest.getSublistValue(options)</code>	string	Returns the value of a sublist line item.
Property	<code>ServerRequest.body</code>	string	The server request body
	<code>ServerRequest.files</code>	Object	The server request files.
	<code>ServerRequest.headers</code>	Object	The server request headers.
	<code>ServerRequest.method</code>	enum	The https method for the server request.
	<code>ServerRequest.parameters</code>	Object	The server request parameters.
	<code>ServerRequest.url</code>	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	<code>ServerResponse.addHeader(options)</code>	void	Adds a header to the response
	<code>ServerResponse.getHeader(options)</code>	void	Returns the value of a response header
	<code>ServerResponse.renderPdf(options)</code>	void	Generates and renders a PDF directly to the response
	<code>ServerResponse.sendRedirect(options)</code>	void	Sets the redirect URL by resolving to a NetSuite resource
	<code>ServerResponse.setCdnCacheable(options)</code>	void	Sets CDN caching for a period of time.
	<code>ServerResponse.setHeader(options)</code>	void	Sets the value of a response header.
	<code>ServerResponse.write(options)</code>	void	Writes information (text/xml/html) to the response.
	<code>ServerResponse.writeFile(options)</code>	void	Writes a file to the response.
	<code>ServerResponse.writeLine(options)</code>	void	Writes line information (text/xml/html) to the response.
	<code>ServerResponse.writePage(options)</code>	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/runtime', 'N/https', 'N/crypto'],  
    function(http, https, crypto) {  
        function createSecureString() {  
            var passwordGuid = '{284CFB2D225B1D76FB94D150207E49DF}';  
            var secureToken = https.createSecureString({  
                input: passwordGuid  
            });  
            var secretKey = https.createSecretKey({  
                input: passwordGuid  
            });  
            secureToken = secureToken.hmac({  
                algorithm: crypto.HashAlg.SHA256,  
                key: secretKey  
            });  
            createSecureString();  
        };  
    });
```

The following example is a Suitelet sample.

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  
 * @ScriptType Suitelet  
 */  
define(['N/ui', '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
    });
    form.addSubmitButton();
    option.response.writePage(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.
Supported Script Types	Server-side scripts
Module	N/https Module
Since	Version 2015 Release 2

SecureString.appendSecureString(options)

Method Description	Method used to append a passed in https.SecureString to another 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.token	https.SecureString object	required	The https.SecureString to append.

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.encoding	string	required	The encoding to apply to the returned string.

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	string	required	The encoding to apply to the returned string.

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.

SecureString.hmac(options)

Method Description	Produces the securestring as an hmac.
Returns	https.SecureString
Supported Script Types	Server-side scripts
Governance	None
Module	N/https Module
Since	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 by the crypto.Hash enum.

Parameter	Type	Required / Optional	Description
options.key	crypto.SecretKey	required	A key returned from https.createSecureKey(options) .

https.createSecureKey(options)

Method Description	Creates and returns a crypto.SecretKey object. You can put the key in your secure string. SuiteScript decrypts the value (key) and sends it to the server
Returns	crypto.SecretKey
Supported Script Types	Server-side scripts
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.encoding	string	required	Specifies the encoding for the SecureKey.	Version 2015 Release 2
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

https.createSecureKey.promise(options)

Method Description	Creates and returns a crypto.SecretKey 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

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
options.inputEncoding	string	optional	Identifies the encoding that the input string uses. The default value is <code>UTF_8</code>	Release 15 Version 2

https.createSecureString.promise(options)

Method Description	Creates and returns an https.SecureString asynchronously.
	Note: For information about the parameters and errors thrown for this method, see https.createSecureString(options) . For additional information on promises, see Promise object .
Returns	<code>SecureTokenResolver</code>
Synchronous Version	https.createSecureString(options)
Supported Script Types	All client-side scripts
Governance	None
Module	N/https Module
Since	Version 2015 Release 2

https.ClientResponse

Object Description	Encapsulates the result of an https request from an HTTPS client. This object is read-only.
Supported Script Types	Server-side scripts
Module	N/https Module
Since	Version 2015 Release 2

ClientResponse.body

Property Description	The 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.

ClientResponse.code

Property Description	The 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.

ClientResponse.headers

Property Description	The response 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.

ClientResponse.method

Property Description	The name of the server request HTTPS method. 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.

ClientResponse.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.

ClientResponse.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.

https.ServerRequest

Object Description	Encapsulates the incoming https request information for an HTTPS server. This object is read-only.
Supported Script Types	Server-side scripts
Module	N/https Module
Since	Version 2015 Release 2

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.

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 line number.	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.

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.

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.

ServerRequest.headers

Property Description	The server request 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.

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.

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.

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.

https.ServerResponse

Object Description	Encapsulates the response to an incoming https request for an HTTPS server.
Supported Script Types	Server-side scripts
Module	N/https Module
Since	Version 2015 Release 2

ServerResponse.addHeader(options)

Method Description	Method used to add a header to the response. If the same header has already been set, this method adds another line for that header. For example:
	Vary: 'Accept-Language' Vary: 'Accept-Encoding'
Returns	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.

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.

ServerResponse.sendRedirect(options)

Method Description	Method used to set the redirect URL by resolving to a NetSuite resource.
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: • RECORD	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> TASKLINK SUITELET 	
options.identifier	string	required	<p>The primary ID for this resource.</p> <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 SUITLET, input the script ID. 	Version 2015 Release 2
options.id	string	optional	<p>The secondary ID for this resource. If the base type is SUITLET, input the deployment ID.</p>	Version 2015 Release 2
options.editMode	boolean true false	optional	If the base type is RECORD, this value determines whether to return a URL for the record in edit mode or view mode.	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.	<p>The value input for options.type is not one of the following:</p> <ul style="list-style-type: none"> RECORD TASKLINK SUITELET
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.

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.

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.

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.

ServerResponse.write(options)

Method Description	Method used to write information (text/xml/html) 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 or file 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.

Error Code	Message	Thrown If
WRONG_PARAMETER_TYPE	{param name}	The value input for options.output is not a string.

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.	

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.

ServerResponse.writeLine(options)

Method Description	Method used to write line information (text/xml/html) 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.

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.F	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.

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:
	{Vary: ['Accept-Language', 'Accept-Encoding']}
Module	N/https Module
Since	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
READ_ONLY_PROPERTY		You attempt to edit this property.

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

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.delete(options)

Method Description	Method used to send an HTTPS DELETE request and returns the response.
	Note: This method does not include an <code>options.body</code> parameter. Postdata is not required when the HTTPS method is a DELETE request.
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
<code>options.url</code>	string	required	The HTTPS URL being requested	Version 2015 Release 2
<code>options.headers</code>	object	optional	The HTTPS headers.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
<code>SSS_MISSING_REQD_ARGUMENT</code>	Missing a required argument: {param name}	A required parameter is not passed.

Syntax

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.request(options)

Method Description	Method used to send an HTTPS request and return the response.
Returns	An <code>https.ClientResponse</code> 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.method	enum	required	The HTTPS request method. Set using the https.Method enum.	Version 2015 Release 2
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. 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

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.post(options)

Method Description	Method used to send an HTTPS PUT request and return server response.
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

Parameter	Type	Required / Optional	Description	Since
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

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.put(options)

Method Description	Method used to send an HTTPS PUT request and return server response.
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

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.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.
	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

- LONG
- MEDIUM
- SHORT
- UNIQUE

Syntax

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	Holds the string values for supported HTTPS requests. This enum is used to set the value of the <code>ClientResponse.method</code> and <code>ServerRequest.method</code> properties.
	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

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](#).

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](#).

- [N/log Module Members](#)
- [N/log Module Guidelines](#)
- [Using Log Levels](#)
- [Viewing Script Execution Logs](#)

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 given 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.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.

...

```
var var1 = "value";
log.audit("Audit Entry", "Value of var1 is: " + var1);
...
```

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.

```
...
```

```
var var1 = "value";
log.debug("Debug Entry", "Value of var1 is: " + var1);
...
```

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, <code>JSON.stringify(obj)</code> 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.

```
...
var var1 = "value";
log.emergency("Emergency Entry", "Value of var1 is: " + var1);
```

...

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.

```
...
var var1 = "value";
log.error("Error Entry", "Value of var1 is: " + var1);
...
```

N/plugin Module

Load the N/plugin module to load custom plug-in implementations.

N/plugin Module Members

Member Type	Name	Return Type / Value Type	Description
Method	plugin.findImplementations(options)	string[]	Returns the script IDs of given custom plug-in type implementations.
Method	plugin.loadImplementation(options)	Object	Instantiates an implementation of the given custom plug-in type.

N/plugin Module Script Sample

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.

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 given custom plug-in type implementations. Returns an empty list when there is no custom plug-in type with the given 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
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

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

```
...
var impls = plugin.findImplementations({
    type: 'customscript_sample_plugin'
```

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

plugin.loadImplementation(options)

Method Description	Instantiates an implementation of the given custom plugin type. Returns the implementation which is currently selected in the UI (Manage Plug-ins page) when no implementation ID is explicitly given.
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 given plug-in type, or the plug-in type does not exist.

Syntax

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

```
...  
var pl = plugin.loadImplementation({  
    type: 'customscript_sample_plugin'  
});  
...
```

N/portlet Module

Load the portlet module to resize or refresh a form portlet. See [Portlet Script Type](#).

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
 */
define([], function () {
    function render(context) {
        var portletObj = context.portlet;
        portletObj.title = 'Portlet API Demo';

        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: ";
            inlineHTMLField.defaultValue += DEFAULT_HEIGHT + "px; width: " + DEFAULT_WIDTH +
            "px'></div>";
            inlineHTMLField.layoutType = 'normal';
            inlineHTMLField.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: 'button',
                label: 'Resize Portlet',
                onClick: 'resize(); refresh();'});
        }
    }
});
```

```

        type: 'inlinehtml',
        label: 'Resize link'});
resizeLink.defaultValue = "<a onclick=\"require(['SuiteScripts/portletApiTestHelper'], \"";
resizeLink.defaultValue += "function(portletApiTestHelper) {portletApiTestHelper.resizePortlet(); }) \"";
resizeLink.defaultValue += "\" 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 onclick=\"require(['SuiteScripts/portletApiTestHelper'], \"";
    refreshLink.defaultValue += "function(portletApiTestHelper) {portletApiTestHelper.refreshPortlet(); }) \"";
    refreshLink.defaultValue += "\" href='#'>Refresh</a>";
}
}

// 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

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.resize();  
...
```

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 when you want 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 field properties on a record.

SuiteScript supports custom records created using SuiteBuilder and standard NetSuite records.

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](#)
- [Field Object Members](#)

- Record Object Members
- N/record Module Script Samples

N/record Module Members

Member Type	Name	Return Type / Value Type	Description
Object	record.Field	Object	Encapsulates a body or sublist field on a standard or custom record.
	record.Record	Object	Encapsulates a NetSuite record.
Method	record.attach(options)	void	Attaches a record to another record.
	record.attach.promise(options)	void	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)	record.Record	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)	record.Record	Creates a new record asynchronously.
	record.delete(options)	number	Deletes a record.
	record.delete.promise(options)	number	Deletes a record asynchronously.
	record.detach(options)	void	Detaches a record.
	record.detach.promise(options)	void	Detaches a record asynchronously.
	record.load(options)	record.Record	Loads an existing record.
	record.load.promise(options)	record.Record	Loads an existing record asynchronously.
	record.submitFields(options)	number	Updates and submits one or more body fields on an existing record in NetSuite. Returns the internal id of the parent record.
	record.submitFields.promise(options)	number	Updates and submits one or more body fields asynchronously on an existing record in NetSuite. Returns the internal id of the parent record.
	record.transform(options)	record.Record	Transforms a record into another record type, using data from an existing record.
	record.transform.promise(options)	record.Record	Transforms a record into another record type asynchronously, using data from an existing record.

Member Type	Name	Return Type / Value Type	Description
Enum	record.Type	enum	Enumeration that holds the string values for supported record types.

Field Object Members

Member Type	Name	Return Type / Value Type	Description
Method	field.getSelectOptions(object)	array	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.
	field.toString()	string	Determines an object's type.
Property	field.label	string (read-only)	UI label for a standard or custom field body or sublist field.
	field.id	string (read-only)	Internal ID of a standard or custom body or sublist field.
	field.sublistId	string (read-only)	Internal ID of a standard or custom sublist field.
	field.type	string (read-only)	Type of a standard or custom body or sublist field.
	field.isMandatory	boolean true false	Returns <code>true</code> if the standard or custom field is mandatory on the record form, or <code>false</code> otherwise. <ul style="list-style-type: none"> Body fields. Available in all script types. You can read or write to this property. Sublist fields. Sublist fields in Client scripts only. In addition, this property is read-only for sublist fields.
	field.isDisabled	boolean true false	Returns <code>true</code> if the standard or custom field is disabled on the record form, or <code>false</code> otherwise. This property is available for body and sublist fields in Client scripts only. You can read or write to this property.
	field.isPopup	boolean true false	Returns <code>true</code> if the field is a popup list field, or <code>false</code> otherwise. This property is available for body and sublist fields in Client scripts only.
	field.isDisplay	boolean true false	Returns <code>true</code> if the field is set to display on the record form. Returns <code>false</code> if the field is set to not display on the record form.

Member Type	Name	Return Type / Value Type	Description
			This property is available in Client scripts only. In addition, it is read-only for sublist fields.
	field.isVisible	boolean true false	Returns <code>true</code> if the field is visible on the record form, and <code>false</code> otherwise. This property is read-only and is available in Client scripts only.
	field.isReadOnly	boolean true false	Returns <code>true</code> if the field on the record form cannot be edited, and <code>false</code> otherwise. This property is read-only and is available in Client scripts only.

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)	object	Cancels the currently selected line on the indicated sublist.
	Record.commitLine(options)	object	Commits the currently selected line
	Record.findSublistLineWithValue(options)	number	Returns the line number for the first occurrence of a field value in a sublist.
	Record.getCurrentSublistIndex(options)	number	Returns the line number of the currently selected line.
	Record.getCurrentSublistSubrecord(options)	object	Gets the subrecord associated with the sublist field on the current line.
	Record.getCurrentSublistText(options)	number Date string Array	Returns a text representation of the field value in the currently selected line.
	Record.getCurrentSublistValue(sublistName, fieldName)	number Date string Array	Returns the value of a sublist field on the currently selected sublist line.
	Record.getField(options)	object	Returns a field object from a record.
	Record.getFields()	array of strings	Returns the field names (internal ids) of all fields on a record.
	Record.getLineCount(options)	number	Return the line count of sublist

Member Type	Name	Return Type / Value Type	Description
	Record.getSublistField(options)	object	Returns a field object from a record's sublist
	Record.getSublistFields(options)	Array of strings	Returns all the field names in a sublist.
	Record.getSublistSubrecord(options)	Object	Gets the subrecord associated with a sublist field
	Record.getSublistText(options)	string	Returns the value of a sublist field in text representation
	Record.getSublistValue(options)		
	Record.getSubrecord(options)		
	Record.getText(options)		
	Record.getValue(fieldName)		
	Record.insertLine(options)		
	Record.removeCurrentSublistSubrecord(options)		
	Record.removeLine(options)		
	Record.removeSublistSubrecord(options)		
	Record.removeSubrecord(options)		
	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)		Selects an existing line in a sublist.
	Record.selectNewLine(options)		Inserts and selects a new line in a sublist.
	Record.setCurrentSublistText(options)		
	Record.setCurrentSublistValue(options)		
	Record.setSublistText(options)		
	Record.setSublistValue(options)		
	Record.setText(options)		
	Record.setValue(fieldName, value)		
	Record.toString()	string	Determines an object's type.

Member Type	Name	Return Type / Value Type	Description
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.

N/record Module Script Samples

The following example shows how to create and save a contact record.

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'],
    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();
    }
);
```

```
});
```

This example shows how to access sublists and a subrecord from a record. This example requires the Advanced Number Inventory Management feature.

```
/**  
 * @NApiVersion 2.x  
 */  
require(['N/record'],  
    function(record) {  
        function createPurchaseOrder() {  
            var rec = record.create({  
                type: 'purchaseorder',  
                isDynamic: true  
            });  
            rec.setValue({  
                fieldId: 'entity',  
                value: 52  
            });  
            rec.setValue({  
                fieldId: 'location',  
                value: 2  
            });  
            rec.selectNewLine({  
                sublistId: 'item'  
            });  
            rec.setCurrentSublistValue({  
                sublistId: 'item',  
                fieldId: 'item',  
                value: 190  
            });  
            rec.setCurrentSublistValue({  
                sublistId: 'item',  
                fieldId: 'quantity',  
                value: 2  
            });  
            subrecordInvDetail = rec.getCurrentSublistSubrecord({  
                sublistId: 'item',  
                fieldId: 'inventorydetail'  
            });  
            subrecordInvDetail.selectNewLine({  
                sublistId: 'inventoryassignment'  
            });  
            subrecordInvDetail.setCurrentSublistValue({  
                sublistId: 'inventoryassignment',  
                fieldId: 'receiptinventorynumber',  
                value: 'myinventoryNumber'  
            });  
            subrecordInvDetail.commitLine({  
                sublistId: 'inventoryassignment'  
            });  
            subrecordInvDetail.selectLine({  
                sublistId: 'inventoryassignment',  
                line: 0  
            });  
            var myInventoryNumber = subrecordInvDetail.getCurrentSublistValue({  
                sublistId: 'inventoryassignment',  
                fieldId: 'receiptinventorynumber'  
            });  
        }  
    }  
);
```

```

    }
    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.copy(options)`
- `record.transform(options)`

The following table lists initialization types that are available to certain SuiteScript-supported records and the values they can contain.

Record	Initialization Type	Values
All SuiteScript-supported records that support form customization.	customform	<customformid>
Assembly Build	assemblyitem	<assemblyitemid>
Cash Refund	entity	<entityid>
Cash Sale	entity	<entityid>
Check	entity	<entityid>
Credit Memo	entity	<entityid>
Customer Payment	entity	<entityid> <invoiceid>
Customer Refund	entity	<entityid>
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>

Record	Initialization Type	Values
		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
AN	Netherlands Antilles
AO	Angola
AQ	Antarctica
AR	Argentina
AS	American Samoa
AT	Austria
AU	Australia
AW	Aruba
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BD	Bangladesh

Country Code	Country Name
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BH	Bahrain
BI	Burundi
BJ	Benin
BL	Saint Barthélemy
BM	Bermuda
BN	Brunei Darussalam
BO	Bolivia
BR	Brazil
BS	Bahamas
BT	Bhutan
BV	Bouvet Island
BW	Botswana
BY	Belarus
BZ	Belize
CA	Canada
CC	Cocos (Keeling) Islands
CD	Congo, Democratic People's Republic
CF	Central African Republic
CG	Congo, Republic of
CH	Switzerland
CI	Cote d'Ivoire
CK	Cook Islands
CL	Chile
CM	Cameroon
CN	China
CO	Colombia
CR	Costa Rica
CU	Cuba
CV	Cap Verde
CX	Christmas Island
CY	Cyprus
CZ	Czech Republic

Country Code	Country Name
DE	Germany
DJ	Djibouti
DK	Denmark
DM	Dominica
DO	Dominican Republic
DZ	Algeria
EC	Ecuador
EE	Estonia
EG	Egypt
EH	Western Sahara
ER	Eritrea
ES	Spain
ET	Ethiopia
FI	Finland
FJ	Fiji
FK	Falkland Islands (Malvina)
FM	Micronesia, Federal State of
FO	Faroe Islands
FR	France
GA	Gabon
GB	United Kingdom (GB)
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

Country Code	Country Name
GU	Guam
GW	Guinea-Bissau
GY	Guyana
HK	Hong Kong
HM	Heard and McDonald Islands
HN	Honduras
HR	Croatia/Hrvatska
HT	Haiti
HU	Hungary
ID	Indonesia
IE	Ireland
IL	Israel
IM	Isle of Man
IN	India
IO	British Indian Ocean Territory
IQ	Iraq
IR	Iran (Islamic Republic of)
IS	Iceland
IT	Italy
JE	Jersey
JM	Jamaica
JO	Jordan
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KH	Cambodia
KI	Kiribati
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

Country Code	Country Name
LB	Lebanon
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
LY	Libyan Arab Jamahiriya
MA	Morocco
MC	Monaco
MD	Moldova, Republic of
ME	Montenegro
MF	Saint Martin
MG	Madagascar
MH	Marshall Islands
MK	Macedonia
ML	Mali
MM	Myanmar
MN	Mongolia
MO	Macau
MP	Northern Mariana Islands
MQ	Martinique
MR	Mauritania
MS	Montserrat
MT	Malta
MU	Mauritius
MV	Maldives
MW	Malawi
MX	Mexico
MY	Malaysia
MZ	Mozambique
NA	Namibia
NC	New Caledonia

Country Code	Country Name
NE	Niger
NF	Norfolk Island
NG	Nigeria
NI	Nicaragua
NL	Netherlands
NO	Norway
NP	Nepal
NR	Nauru
NU	Niue
NZ	New Zealand
OM	Oman
PA	Panama
PE	Peru
PF	French Polynesia
PG	Papua New Guinea
PH	Philippines
PK	Pakistan
PL	Poland
PM	St. Pierre and Miquelon
PN	Pitcairn Island
PR	Puerto Rico
PS	Palestinian Territories
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

Country Code	Country Name
SE	Sweden
SG	Singapore
SH	St. Helena
SI	Slovenia
SJ	Svalbard and Jan Mayen Islands
SK	Slovak Republic
SL	Sierra Leone
SM	San Marino
SN	Senegal
SO	Somalia
SR	Suriname
ST	Sao Tome and Principe
SV	El Salvador
SY	Syrian Arab Republic
SZ	Swaziland
TC	Turks and Caicos Islands
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

Country Code	Country Name
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	Western Samoa
YE	Yemen
YT	Mayotte
ZA	South Africa
ZM	Zambia
ZW	Zimbabwe

record.Field

Object Description	Object that 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)• currentRecord object in Client script types
Supported Script Types	All script types
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](#).

```
...
var rec = record.load({type:'inventoryitem',id:'43'}); // inventory item with internal id of 43
var field = rec.getField('itemid'); // get Field object
...
```

{}

field.getSelectOptions(object)

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. This method returns an array in the following format: [{value: 5, text: abc}, {value: 6, text: 123}] Returns <code>Type Error</code> if the field is not a select field. You can only use this method on a record in dynamic mode.
Returns	array
Supported Script Types	All script types
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.filter	string	Optional	<p>Search string to filter the select options that are returned.</p> <p>If there are 50 select options available, and 10 of the options contains John, for example, John Smith or Shauna Johnson, only the 10 options will be returned.</p> <p>Note: Filter values are case insensitive. The filters John and john will return the same select options.</p>	Version 2015 Release 2
options.operator	string	Optional	Supported operators are contains is startswith . If not specified, defaults to the contains operator.	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](#).

```
...
var rec = record.load({
    type:'opportunity',
    id:'333',
    isDynamic: true});
var field = rec.getField('entity');
var options = field.getSelectOptions('C', 'startswith'); // get all entity options that start
with 'C'
...
```

field.label

Property Description	UI label for a standard or custom field body or sublist field.
Type	string
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](#).

```
...
var rec = record.load({type:'inventoryitem',id:'43'});
var field = rec.getField('itemid');
var label = field.label;
...
```

field.id

Property Description	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

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](#).

```
...
var rec = record.load({type:'inventoryitem',id:'43'});
var field = rec.getField('itemid');
var id = field.id;
...
```

field.sublistId

Property Description	Internal ID of a standard or custom sublist field.
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](#).

```
...
var rec = record.load({type:'salesorder',id:'43'});
var sublistField = rec.getSublistField({sublistId: 'inventoryassignment',
    fieldId: 'receiptinventorynumber',
    value: 'myinventoryNumber'
});
var id = sublistField.sublistId;
...
```

field.type

Property Description	Type of a standard or custom body or sublist field. For example, value can be text, date, currency, select, checkbox, etc.
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](#).

```
...
var rec = record.load({type:'inventoryitem',id:'43'});
var field = rec.getField('itemid');
var type = field.type;
...
```

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.
	This property is available for the following field types:

- Body fields. Available in all script types. You can read or write to this property.
- Sublist fields. Sublist fields in Client scripts only. In addition, this property is read-only for sublist fields.

Type	boolean true false
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](#).

```
...
var rec = record.load({type:'inventoryitem',id:'43'});
var field = rec.getField('itemid');
if (field.isMandatory)
{
    // body field 'itemid' is a mandatory body field
}
...
```

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. This property is available for body and sublist fields in Client scripts only. You can read or write to this property.
Type	boolean true false
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](#).

```
...
var field = currentRecord.getField('memo');
if (field.isDisabled)
{
    // field is disabled on the record form
}
...
```

field.isPopup

Property Description	Returns <code>true</code> if the field is a popup list field, or <code>false</code> otherwise.
-----------------------------	--

	This property is available for body and sublist fields in Client scripts only.
Type	boolean (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](#).

```
...
var field = currentRecord.getField('entity');
if (field.isPopup)
{
    // field is a popup list field
}
...
```

field.isDisplay

Property Description	Returns <code>true</code> if the field is set to display on the record form. Returns <code>false</code> if the field is set to not display on the record form. Fields can be a part of a record even if they do not display on the record form. This property is available in Client scripts only. In addition, it is read-only for sublist fields.
Type	boolean <code>true</code> <code>false</code>
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](#).

```
...
var field = currentRecord.getField('memo');
if (field.isDisplay)
{
    // field is set to be displayed on the record form
}
...
```

field.isVisible

Property Description	Returns <code>true</code> if the field is visible on the record form, and <code>false</code> otherwise.
-----------------------------	---

	This property is read-only and is available in Client scripts only.
Type	boolean true false
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](#).

```
...
var field = currentRecord.getField('memo');
if (!field.isVisible)
{
    // field is invisible on the record form
}
...
```

field.isReadOnly

Property Description	Returns <code>true</code> if the field on the record form cannot be edited, and <code>false</code> otherwise. This property is read-only and is available in Client scripts only.
Type	boolean true false
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](#).

```
...
var field = currentRecord.getField('memo');
if (field.isReadOnly)
{
    // field is read-only on the record form
}
...
```

record.Record

Load the record module when you want to work with NetSuite records.

Object Description	Encapsulates a NetSuite record.
---------------------------	---------------------------------

There are two modes you can operate in when you create, copy, load, or transform a record with SuiteScript 2.0: standard mode and dynamic mode.

- When a SuiteScript 2.0 script creates, copies, loads, or transforms a record in standard mode, the record's body fields and sublist line items are not sourced, calculated, and validated until the record is saved (submitted) with [Record.save\(options\)](#).

When you work with a record in standard mode, you do not need to set values in any particular order. Once the record is submitted, 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 one of 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.

Supported Script Types	All client and server-side scripts
Module	N/record Module
Since	Version 2015 Release 2

Syntax

```
...
function(record){
    var recObj = record.load({type: record.Type.SALES_ORDER,
                            id: '6',
                            isDynamic: true});

    ...
    recObj.save();
}
...

```

Record.cancelLine(options)

Method Description	Method used to cancel the currently selected line on the indicated sublist.
Returns	The record.Record object that called the method.
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.

Syntax

```
...
    function(record){
        var recObj = record.load({type: record.Type.SALES_ORDER,
                                  id: 6});
        recObj.selectNewLine({sublistId: 'item'});
        recObj.cancelLine({sublistId: 'item'});
    }
...

```

Record.commitLine(options)

Method Description	Method used to commit the currently selected line.
Returns	The record.Record object that called the method.
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.
SSS_INVALID_SUBLIST_OPERATION		The passed sublist ID is invalid.

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](#).

```
...
    var recObj = record.load({type: 'salesorder',
                             id: 6});
    recObj.selectNewLine({sublistId: 'item'});
    recObj.setCurrentSublistValue('item','amount','10.00');
    recObj.commitLine({sublistId: 'item'});
    recObj.save();
...

```

Record.findSublistLineWithValue(options)

Method Description	Method used to return the line number for the first occurrence of a field value in a sublist.
Returns	A line number as a number. Returns -1 if not found
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<ul style="list-style-type: none"> This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2
options.value	number Date string Array	optional		Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.

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](#).

Record.getCurrentSublistIndex(options)

Method Description	Method used to return the line number of the currently selected line.
Returns	A line number as a number.
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			Working with the SuiteScript Records Browser.	

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.

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](#).

Record.getCurrentSublistSubrecord(options)

Method Description	Method used to get the subrecord associated with the sublist field on the current line.
Returns	A subrecord
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	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](#).

Record.getCurrentSublistText(options)

Method Description	Method used to return a text representation of the field value in the currently selected line.
Returns	The value of the field as a number Date string Array
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.
SSS_INVALID_SUBLIST_OPERATION		A required argument is invalid.

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](#).

Record.getCurrentSublistValue(sublistName, fieldName)

Method Description	Method used to return the value of a sublist field on the currently selected sublist line.
---------------------------	--

	Note: This method returns the native value of a field. For
Returns	Value of the field as a number Date string Array
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.
SSS_INVALID_LIST_OPERATION		A required argument is invalid.

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](#).

Record.getField(options)

Method Description	Method used to return a field object from a record.
Returns	A record.Field object
Supported Script Types	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	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.

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](#).

Record.getFields()

Method Description	Method used to return all fields on a record.
Returns	A string array of field names (field internal IDs).
Supported Script Types	Server-side scripts
Governance	None
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](#).

Record.getLineCount(options)

Method Description	Method used to return the line count of sublist
Returns	The line count as a number
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	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](#).

Record.getSublistField(options)

Method Description	Method used to return a field object from a record's sublist.
Returns	A record.Field object
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			information, see the help topic Working with the SuiteScript Records Browser .	
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2
options.line	boolean true false	required		Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.
SSS_INVALID_SUBLIST_OPERATION		A required argument is not valid.

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](#).

Record.getSublistFields(options)

Method Description	Method used to return all the field names in a sublist.
Returns	A string array of field ids.
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.

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](#).

Record.getSublistSubrecord(options)

Method Description	Method used to get the subrecord associated with a sublist field
Returns	A subrecord as a <code>record.Record</code> object
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2
options.line	number	required		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](#).

Record.getSublistText(options)

Method Description	Method used to return the value of a sublist field in text representation
---------------------------	---

Returns	The value of a sublist field as a string
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2
options.line	number	required		Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.
SSS_INVALID_SUBLIST_OPERATION		A required argument is not valid.
SSS_INVALID_API_USAGE		

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](#).

Record.getSublistValue(options)

Method Description	Method used to return the value of a sublist field.
Returns	The value of a sublist field as a number Date string Array
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2
options.line	number	required		Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.
SSS_INVALID_SUBLIST_OPERATION		A required argument is not valid.
SSS_INVALID_API_USAGE		

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](#).

Record.getSubrecord(options)

Method Description	Method used to get the subrecord for the associated field.
Returns	A subrecord
Supported Script Types	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	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.
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

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](#).

Record.getText(options)

Method Description	Method used to get the value of the field in text representation
Returns	The field value as a string
Supported Script Types	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	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.

Error Code	Message	Thrown If
SSS_INVALID_API_USAGE		

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](#).

Record.getValue(fieldName)

Method Description	Method used to return the value of a field.
Returns	The field value as a number Date string Array
Supported Script Types	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	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.
SSS_INVALID_API_USAGE		

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](#).

Record.insertLine(options)

Method Description	Method used to insert a sublist line.
Returns	A record.Record object
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2
options.line	number	required		Version 2015 Release 2
options.ignoreRecalc	boolean true false	required	If set to true, recalc scripting is ignored. The default value is false.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.
SSS_INVALID_SUBLIST_OPERATION		The sublistId is invalid or the line is not editable

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](#).

Record.removeCurrentSublistSubrecord(options)

Method Description	Method used to remove the subrecord for the associated sublist field on the current line.
Returns	A record.Record object
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	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](#).

Record.removeLine(options)

Method Description	Method used to remove a sublist line.
Returns	A record.Record object
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.line	number	required		Version 2015 Release 2
options.ignoreRecalc	boolean true false	required	If set to true, recalc scripting is ignored. The default value is false.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.
SSS_INVALID_SUBLIST_OPERATION		The sublistId is invalid or the line is not editable

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](#).

Record.removeSublistSubrecord(options)

Method Description	Method used to remove the subrecord for the associated sublist field.
Returns	A record.Record object
Supported Script Types	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	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.line	number	required		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](#).

Record.removeSubrecord(options)

Method Description	Method used to remove the subrecord for the associated field.
Returns	A record.Record object.
Supported Script Types	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	Internal ID of a standard or custom body or sublist field.	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](#).

Record.save(options)

Method Description	<p>Method used to:</p> <ul style="list-style-type: none"> Save (submit) a new record. Save edits to an existing record. <p>When working with records in standard mode, you must submit and then load the record to obtain sourced, validated, and calculated field values.</p> <p>Note: This method has an asynchronous counterpart you can use with client scripts. See Record.save.promise(options).</p>
Returns	The internal ID of the new or updated record. This value is returned as a number.

Supported Script Types	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 true false	optional	<ul style="list-style-type: none"> If set to true, sources dependent field information for empty fields. Defaults to false – dependent field values are not sourced. <p>Note: This parameter applies only to standard records. When working with records in dynamic mode, field values are always sourced and the value you provide for enableSourcing is ignored.</p>	Version 2015 Release 2
options.ignoreMandatoryFields	boolean true false	optional	<ul style="list-style-type: none"> Disables mandatory field validation for this save operation. If set to true, ignores all standard and custom fields that were made mandatory through customization. All fields that were made mandatory through company preferences are also ignored. Defaults to false. <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>	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](#).

```
...
var recordId = recordObj.save({
    enableSourcing: false,
    ignoreMandatoryFields: false
});
...

```

Record.save.promise(options)

Method Description	<p>Method used to:</p> <ul style="list-style-type: none"> Save (submit) a new record asynchronously. Save edits to an existing record asynchronously. <p>Note: For information about the parameters and errors thrown for this method, see Record.save(options). For additional information on promises, see Promise object.</p>
Returns	The internal ID of the new or updated record. This value is returned as a number.
Synchronous Version	Record.save(options)
Supported Script Types	All 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

Record.selectLine(options)

Method Description	Method used to select an existing line in a sublist.
Returns	A record.Record object
Supported Script Types	Server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			information, see the help topic Working with the SuiteScript Records Browser .	
options.line	number	required		Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.
SSS_INVALID_SUBLIST_OPERATION		The sublistId is invalid or the line is not editable

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](#).

Record.selectNewLine(options)

Method Description	Method used to insert and select a new line in a sublist.		
Returns	A record.Record object		
Supported Script Types	Server-side scripts		
Governance	None		
Module	N/record Module		
Since	Version 2015 Release 2		

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.

Error Code	Message	Thrown If
SSS_INVALID_SUBLIST_OPERATION		The sublistId is invalid or the line is not editable

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](#).

Record.setCurrentSublistValue(options)

Method Description	Method used to set the value for field in the current selected line.
Returns	A record.Record object
Supported Script Types	Server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2
options.value	number Date string Array	required		Version 2015 Release 2
options.ignoreFieldChange	boolean true false	required	If set to true, the field change and slaving event is ignored. The default value is false.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.
A_SCRIPT_IS_ATTEMPTING_TO_EDIT_THE_1_SUBLIST_THIS_SUBLIST_IS_CURRENTLY_IN_READONLY_MODE_AND_CANNOT_		A user tries to edit read-only sublist field.

Error Code	Message	Thrown If
BE_EDITED_CALL_YOUR_NETSUITE_ADMINISTRATOR_TO_DISABLE_THIS_SCRIPT_IF_YOU_NEED_TO_SUBMIT_THIS_RECORD		

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](#).

Record.setCurrentSublistText(options)

Method Description	Method used to set the value for a field in the current selected line			
Returns	DynamicRecord			
Supported Script Types	Server-side scripts			
Governance	None			
Module	N/record Module			
Since	Version 2015 Release 2			

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2
options.text	number Date string Array	required		Version 2015 Release 2
options.ignoreFieldChange	boolean true false	required	If set to true, the field change and slaving event is ignored. The default value is false.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.
A_SCRIPT_IS_ATTEMPTING_TO_EDIT_THE_1_SUBLIST_THIS_SUBLIST_IS_CURRENTLY_IN_READONLY_MODE_AND_CANNOT_		A user tries to edit read-only sublist field.

Error Code	Message	Thrown If
BE_EDITED_CALL_YOUR_NETSUITE_ADMINISTRATOR_TO_DISABLE_THIS_SCRIPT_IF_YOU_NEED_TO_SUBMIT_THIS_RECORD		

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](#).

Record.setSublistText(options)

Method Description	Method used to set the value of a sublist field.
Returns	A record.Record
Supported Script Types	Server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2
options.line	number	required		Version 2015 Release 2
options.text	string	required		Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.
SSS_INVALID_SUBLIST_OPERATION		The passed sublist ID, field ID, or line number is invalid.

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](#).

Record.setSublistValue(options)

Method Description	Method used to set the value of a sublist field.
Returns	A record.Record object.
Supported Script Types	Server-side scripts
Governance	None
Module	N/record Module
Since	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.sublistId	string	required	<ul style="list-style-type: none"> The internal ID of the sublist. This value is displayed in the Records Browser. For additional information, see the help topic Working with the SuiteScript Records Browser. 	Version 2015 Release 2
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2
options.line	number	required		Version 2015 Release 2
options.value	number Date string Array	required		Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.
SSS_INVALID_SUBLIST_OPERATION		The passed sublist ID, field ID, or line number is invalid.

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](#).

Record.setText(options)

Method Description	Method used to set the value of the field by text representation.
Returns	A record.Record object
Supported Script Types	Server-side scripts

Governance	None			
Module	N/record Module			
Since	Version 2015 Release 2			

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2
options.text	string	required		Version 2015 Release 2
options.ignoreFieldChange	boolean true false	required	If set to true, the field change and slaving event is ignored. The default value is false.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.

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](#).

Record.setValue(fieldName, value)

Method Description	Method used to set the value of the field			
Returns	A record.Record			
Supported Script Types	Server-side scripts			
Governance	None			
Module	N/record Module			
Since	Version 2015 Release 2			

Parameter	Type	Required / Optional	Description	Since
options.fieldId	string	required	Internal ID of a standard or custom body or sublist field.	Version 2015 Release 2
options.value	number Date string Array	required		Version 2015 Release 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. The default value is false.	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.

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](#).

Record.id

Property Description	The internal ID of a specific record. This property is read-only.
Type	number
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](#).

Record.isDynamic

Property Description	Reflects whether the record is in dynamic or standard mode. <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>. <p>When you work with a record in standard mode, you do not need to set values in any particular order. Once the record is submitted, NetSuite processes the record's body fields and sublist line items in the correct order, regardless of the organization of your script.</p>
-----------------------------	--

- When a SuiteScript 2.0 script creates, copies, loads, or transforms a record in dynamic mode, the record's body fields and sublist line items are sourced, calculated, and validated in real-time. A record in dynamic mode emulates the behavior of a record in the UI.

When you work with a record in dynamic mode, it is important that you set values in the same order you would within the UI. If you fail to do this, your results may not be accurate.

This property is read-only. This value is set when the record is created or accessed.

Type	boolean true false
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](#).

Record.type

Property Description	<p>The record type.</p> <p>This property is read-only. This value is set with the <code>record.Type</code> enum during record creation.</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>
Type	string
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](#).

record.attach(options)

Method Description	Method used to attach a record to another record.
Returns	Void
Supported Script Types	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	<ul style="list-style-type: none"> The record to be attached 	Version 2015 Release 2
options.record.type	object	required	<ul style="list-style-type: none"> The type of record to attach. To attach a file from the file cabinet to a record, set type to file. 	Version 2015 Release 2
options.record.id	number string	required	<ul style="list-style-type: none"> The internal id of the record to attach 	Version 2015 Release 2
options.to	record.Record	required	<ul style="list-style-type: none"> The destination record where options.record will be attached to 	Version 2015 Release 2
options.to.type	string	required	<ul style="list-style-type: none"> The record type of the record to attach to. 	Version 2015 Release 2
options.to.id	number string	required	<ul style="list-style-type: none"> The id of the record to attach to. 	Version 2015 Release 2
options.attributes	Object	optional	<ul style="list-style-type: none"> Name/value pairs containing attributes for the attachment. 	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](#).

record.attach.promise(options)

Method Description	Note: For information about the parameters and errors thrown for this method, see record.attach(options) . For additional information on promises, see Promise object .
Returns	
Synchronous Version	record.attach(options)

Supported Script Types	All client-side scripts
Governance	
Module	N/record Module
Since	Version 2015 Release 2

record.copy(options)

Method Description	Method used to create a new record by copying an existing record in NetSuite.
Returns	A new record.Record
Supported Script Types	All 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	<ul style="list-style-type: none"> The record type. Sets the value for the Record.type property. This property is read-only and cannot be changed after the new record is created. Use the record.Type enum to set the value. 	Version 2015 Release 2
options.id	number	required	<ul style="list-style-type: none"> 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. 	Version 2015 Release 2
options.isDynamic	boolean true false	optional	<ul style="list-style-type: none"> If set to true, the new record is created in dynamic mode. If set to false, the new record is created in standard mode. <ul style="list-style-type: none"> When a SuiteScript 2.0 script creates, copies, loads, 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<p>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).</p> <p>When you work with a record in standard mode, you do not need to set values in any particular order. Once the record is submitted, NetSuite processes the record's body fields and sublist line items in the correct order, regardless of the organization of your script.</p> <ul style="list-style-type: none"> 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. <p>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> <ul style="list-style-type: none"> The default value is false. The default value is null. 	
options.defaultValues	Object	optional		

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT	A required argument is not passed.	

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](#).

```
...
var recObj = record.copy({
    type: record.Type.SALES_ORDER,
    id: 157,
    isDynamic: true,
});
...

```

record.copy.promise(options)

Method Description	Method used to create a new record object asynchronously by copying an existing record object in NetSuite.
	Note: For information about the parameters and errors thrown for this method, see record.copy(options) . For additional information on promises, see Promise object .
Returns	A new record.Record object.
Synchronous Version	record.copy(options)
Supported Script Types	All 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

Syntax

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](#).

```
var recObj = record.copy.promise({
    type: record.Type.SALES_ORDER,
    id: 157,
    isDynamic: true,
});
```

record.create(options)

Method Description	Method used to create a new record.Record Object in NetSuite.
---------------------------	---

	For the promise version of this method, see record.create.promise(options) . Note that promises are only supported in client scripts.
Returns	A new record.Record Object
Supported Script Types	All 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	<ul style="list-style-type: none"> The record type. Sets the value for the Record.type property. This property is read-only and cannot be changed after the record is created. Use the record.Type enum to set the value. 	Version 2015 Release 2
options.isDynamic	boolean true false	optional	<ul style="list-style-type: none"> If set to true, the record is created in dynamic mode. If set to false, the record is created 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 Record.save(options). <p>When you work with a record in standard mode, you do not need to set values in any particular order. Once the record</p>	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<p>is submitted, NetSuite processes the record's body fields and sublist line items in the correct order, regardless of the organization of your script.</p> <ul style="list-style-type: none"> 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. <p>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> <ul style="list-style-type: none"> The default value is false. 	
options.defaultValues	Object	optional	<ul style="list-style-type: none"> The default value is null. 	

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.

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](#).

```
var recObj = record.create({
    type: record.Type.SALES_ORDER,
    isDynamic: true,
    defaultValues: {entity: 87}
});
```

record.create.promise(options)

Method Description	Method used to create a new record object asynchronously in NetSuite.
	<p>Note: For information about the parameters and errors thrown for this method, see record.create(options). For additional information on promises, see Promise object.</p>
Returns	A new record.Record object.
Synchronous Version	record.create(options)
Supported Script Types	All 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

Syntax

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](#).

```
var recObj = record.create.promise({
    type: record.Type.SALES_ORDER,
    isDynamic: true,
    defaultValues: {entity: 87}
});
```

record.delete(options)

Method Description	Method used to delete a record in NetSuite.
Returns	The internal ID of the deleted record.Record
Supported Script Types	All 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.type	string	required	<ul style="list-style-type: none"> The record type. Sets the value for the Record.type property. This property is read-only. Use the record.Type enum to set the value. 	Version 2015 Release 2
options.id	number	required	<ul style="list-style-type: none"> The internal ID of the record instance to be deleted. The internal ID of the record is displayed on the list page for the record type. 	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.

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](#).

```
...
var recObj = record.delete({
    type: record.Type.SALES_ORDER,
    id: 128,
});
...
```

record.delete.promise(options)

Method Description	Method used to delete a record object in NetSuite.
	Note: For information about the parameters and errors thrown for this method, see record.delete(options) . For additional information on promises, see Promise object .
Returns	The internal ID of the deleted record object.
Synchronous Version	record.delete(options)

Supported Script Types	All 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

Syntax

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](#).

```
var recObj = record.delete.promise({
    type: record.Type.SALES_ORDER,
    id: 128,
});
```

record.detach(options)

Method Description	Detaches a record from another record.
Returns	Void
Supported Script Types	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	object	required	The type of record to be detached.	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 where options.record will be detached from.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
options.from.type	string	required	The type of the destination.	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 attributes.	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](#).

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 additional information on promises, see Promise object .
Returns	Void
Synchronous Version	record.detach(options)
Supported Script Types	All client-side scripts
Governance	
Module	N/record Module
Since	Version 2015 Release 2

record.load(options)

Method Description	Method used to load an existing record object in NetSuite.
Returns	An existing record.Record
Supported Script Types	All 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	<ul style="list-style-type: none"> The record type. Sets the value for the Record.type property. This property is read-only and cannot be changed after the record is loaded. Use the record.Type enum to set the value. 	Version 2015 Release 2
options.id	number	required	<ul style="list-style-type: none"> 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	<ul style="list-style-type: none"> If set to true, the record is loaded in dynamic mode. If set to false, the record is loaded 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 Record.save(options). When you work with a record in standard mode, you do not need to set values in any particular order. Once the record is submitted, 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 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<p>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.</p> <p>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> <ul style="list-style-type: none"> The default value is false. 	
options.defaultValues	Object	optional	<ul style="list-style-type: none"> The default value is null. 	

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.

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](#).

```
...
var recObj = record.load({
    type: record.Type.SALES_ORDER,
    id: 157,
    isDynamic: true,
});
...
```

record.load.promise(options)

Method Description	Method used to load an existing record object in NetSuite.
Note:	For information about the parameters and errors thrown for this method, see record.load(options) . For additional information on promises, see Promise object .
Returns	An existing record.Record object.

Synchronous Version	record.load(options)
Supported Script Types	All 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

Syntax

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](#).

```
var recObj = record.load.promise({
    type: record.Type.SALES_ORDER,
    id: 157,
    isDynamic: true,
});
```

record.submitFields(options)

Method Description	Method used to update and submit one or more body fields on an existing record in NetSuite. 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 additional information, see the help topic Using Inline Editing.• Custom body fields that support inline editing. You cannot use this method to edit and submit the following: <ul style="list-style-type: none">• Select fields• Sublist line item fields• Subrecord fields (for example, address fields)
Returns	The internal ID of the parent record.
Supported Script Types	All 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	<ul style="list-style-type: none"> The record type. Sets the value for the <code>Record.type</code> property. This property is read-only and cannot be changed after the record is loaded. Use the <code>record.Type</code> enum to set the value. 	Version 2015 Release 2
options.id	number string	required	<ul style="list-style-type: none"> 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. 	Version 2015 Release 2
options.values	Object	required	<ul style="list-style-type: none"> An object of field ID / field value pairs for each field you want to edit and submit. 	Version 2015 Release 2
options.options	Object	optional		Version 2015 Release 2
options.defaultValue	Object	optional	<ul style="list-style-type: none"> The default value is null. 	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](#).

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

```

record.submitFields.promise(options)

Method Description	Method used to update and submit one or more body fields asynchronously on an existing record in NetSuite. When you use this method, you do not need to load or submit the parent record object.
---------------------------	--

	Note: For information about the parameters and errors thrown for this method, see record.submitFields(options) . For additional information on promises, see Promise object .
Returns	The internal ID of the parent record.
Synchronous Version	record.submitFields(options)
Supported Script Types	All 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

record.transform(options)

Method Description	Method used to transform a record into another record type, 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	A record.Record that encapsulates the transformed 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.type	string	required	<ul style="list-style-type: none"> The record type of the existing record instance being transformed. Sets the value for the Record.type property. This property is read- 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<p>only and cannot be changed after the record is loaded.</p> <ul style="list-style-type: none"> • Use the record.Type enum to set the value. 	
options.id	number	required	<ul style="list-style-type: none"> • The internal ID of the existing record instance being transformed. The internal ID of the record is displayed on the list page for the record type. 	Version 2015 Release 2
options.toType	string	required	<ul style="list-style-type: none"> • The record type of the record returned once the transformation is complete. 	Version 2015 Release 2
options.isDynamic	boolean true false	optional	<ul style="list-style-type: none"> • If set to true, the new record is created in dynamic mode. If set to false, the new record is created 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 Record.save(options). When you work with a record in standard mode, you do not need to set values in any particular order. Once the record is submitted, 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 	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<p>in real-time. A record in dynamic mode emulates the behavior of a record in the UI.</p> <p>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> <ul style="list-style-type: none"> The default value is false. 	
options.defaultValues	Object	optional	<ul style="list-style-type: none"> Defaults that are passed upon record initialization. 	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_MISSING_REQD_ARGUMENT		A required argument is not passed.

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](#).

```
var recObj = record.transform({
    fromType: record.Type.CUSTOMER,
    fromId: 107,
    toType: record.Type.SALES_ORDER,
    isDynamic: true,
});
```

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

Original Record Type	Transformed Record Type
Customer	Opportunity
Customer	Sales Order
Employee	Expense Report
Employee	Time
Quote	Cash Sale
Quote	Invoice
Quote	Sales Order
Invoice	Credit Memo
Invoice	Customer Payment
Invoice	Return Authorization
Lead	Opportunity
Opportunity	Cash Sale
Opportunity	Quote
Opportunity	Invoice
Opportunity	Sales Order
Prospect	Quote
Prospect	Opportunity
Prospect	Sales Order
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

Original Record Type	Transformed Record Type
Vendor	Vendor Bill
Vendor Bill	Vendor Credit
Vendor Bill	Vendor Payment
Vendor Bill	Vendor Return Authorization
Vendor Return Authorization	Item Fulfillment
Vendor Return Authorization	Vendor Credit
Work Order	Assembly Build

record.transform.promise(options)

Method Description	Method used to transform a record into another record type asynchronously, using data from an existing record.
	<p>Note: For information about the parameters and errors thrown for this method, see record.transform(options). For additional information on promises, see Promise object.</p>
Returns	A record.Record object that encapsulates the transformed record.
Synchronous Version	record.transform(options)
Supported Script Types	All 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

Syntax

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](#).

```
var recObj = record.transform.promise({
    fromType: record.Type.CUSTOMER,
    fromId: 107,
    toType: record.Type.SALES_ORDER,
    isDynamic: true,
});
```

record.Type

Enum Description	Enumeration that holds the string values for supported record types. This enum is used to set the value of the Record.type property. Note that the Record.type . property is read only. Its value must be set using this enum. 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/record Module
Since	Version 2015 Release 2

Values

<ul style="list-style-type: none"> • ACCOUNT • ACCOUNTING_BOOK • ADDRESS • AMORTIZATION_SCHEDULE • AMORTIZATION_TEMPLATE • ACTIVITY • ASSEMBLY_BUILD • ASSEMBLY_UNBUILD • BILLING_CLASS • BILLING_SCHEDULE • BIN • BIN_PUTAWAY_WORKSHEET • BIN_TRANSFER • BLANKET_PURCHASE_ORDER • BUILD_ASSEMBLY • CAMPAIGN • CAMPAIGN_TEMPLATE • CASE • CASH_REFUND • CASH_SALE • CHARGE • CHECK • CLASS • COMPETITOR 	<ul style="list-style-type: none"> • GIFT_CERTIFICATE_ITEM • GLOBAL_ACCOUNT_MAPPING • GROUP • INTERCOMPANY_JOURNAL_ENTRY • INVENTORY_ADJUSTMENT • INVENTORY_COST_REVALUATION • INVENTORY_COUNT • INVENTORY_DETAIL • INVENTORY_ITEM • INVENTORY_NUMBER • INVENTORY_TRANSFER • INVOICE • ISSUE • ITEM_ACCOUNT_MAPPING • ITEM_SEARCH • ITEM_DEMAND_PLAN • ITEM_FULFILLMENT • ITEM_GROUP • ITEM_RECEIPT • ITEM_REVISION • ITEM_SUPPLY_PLAN • JOURNAL_ENTRY 	<ul style="list-style-type: none"> • PROJECT_TASK • PROMOTION • PROSPECT • PURCHASE_CONTRACT • PURCHASE_ORDER • REALLOCATE_ITEMS • REQUISITION • RESOURCE_ALLOCATION • RETURN_AUTHORIZATION • REVENUE_COMMITMENT • REVENUE_COMMITMENT_REVERSAL • REVENUE_RECOGNITION_SCHEDULE • REVENUE_RECOGNITION_TEMPLATE • SALES_ORDER • SALES_TAX_ITEM • SCHEDULED_SCRIPT_INSTANCE • SERIALIZED_ASSEMBLY_ITEM • SERIALIZED_INVENTORY_ITEM • SERVICE • SOLUTION • STATISTICAL_JOURNAL_ENTRY
---	--	--

• CONTACT	• KIT	• SUBSIDIARY
• COUPON_CODE	• LANDED_COST	• SUBTOTAL
• CREDIT_MEMO	• LEAD	• TASK
• CURRENCY	• LOCATION	• TAX_CONTROL_ACCOUNT
• CUSTOMER	• LOT_NUMBERED_ASSEMBLY_ITEM	• TAX_GROUP
• CUSTOMER_CATEGORY	• LOT_NUMBERED_INVENTORY_ITEM	• TAX_PERIOD
• CUSTOMER_DEPOSIT	• MANUFACTURING_COST_TEMPLATE	• TAX_TYPE
• CUSTOMER_PAYMENT	• MANUFACTURING_PLANNED_TIME	• TERM
• CUSTOMER_REFUND	• MANUFACTURING_OPERATION_TASK	• TIME
• CUSTOM_LIST	• MANUFACTURING_ROUTING	• TOPIC
• DEPARTMENT	• MARKUP	• TRANSACTION_SEARCH
• DEPOSIT	• MESSAGE	• TRANSFER_ORDER
• DEPOSIT_APPLICATION	• MULTIBOOK_ACCOUNTING_TRANSACTION	• UNIT_OF_MEASURE
• DESCRIPTION	• NEXUS	• VENDOR
• DISCOUNT	• NON_INVENTORY_PART	• VENDOR_BILL
• DOWNLOAD_ITEM	• NOTE	• VENDOR_CATEGORY
• EMAIL_TEMPLATE	• OPPORTUNITY	• VENDOR_CREDIT
• EMPLOYEE	• OTHER_CHARGE_ITEM	• VENDOR_PAYMENT
• ENTITY	• OTHER_NAME	• VENDOR_RETURN_AUTHORIZATION
• ESTIMATE_QUOTE	• PARTNER	• WEB_SITE_SETUP
• EVENT	• PAYCHECK_JOURNAL	• WORK_ORDER
• EXPENSE_CATEGORY	• PAYMENT	• WORK_ORDER_CLOSE
• EXPENSE_REPORT	• PAYROLL_ITEM	• WORK_ORDER_COMPLETION
• FOLDER	• PHONE_CALL	• WORK_ORDER_ISSUE
• GIFT_CERTIFICATE	• PRICE_LEVEL	
	• PROJECT_JOB	
	• PROJECT_EXPENSE_TYPE	

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](#).

N/redirect Module

Use the redirect module to customize navigation within NetSuite by setting up a redirect URL that resolves to a NetSuite resource or external URL. You can redirect users to one of the following:

- URL
- Suitelet
- Record
- Task link
- Saved search
- Unsaved search

Note: Suitelets, beforeLoad user events, and synchronous afterSubmit user events are supported. Some module members support client-side scripts. 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.
	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.

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

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.url	string	required	The URL of a Suitelet that is available externally Note: For an external URL, Available without Login must be enabled on the Script Deployment page for the Suitelet.
options.parameters	Object	optional	Contains additional URL parameters as key/value pairs.

Syntax

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.redirect({
        url: '/app/site/hosting/scriptlet.nl?script=130&deploy=1',
        parameters: {'custparam_test':'helloWorld'}
    });
...

```

redirect.toRecord(options)

Method Description	Method used to set the redirect URL to a specific NetSuite record.
	Note: If you redirect a user to a record, the record must first exist in NetSuite. If you want to redirect a user to a new record, you must first create and submit the record before redirecting them. You must also ensure that any required fields for the new record are populated before submitting the record.
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 false – returns the URL to a record in view mode.
options.parameters	Object	optional	Contains additional URL parameters as key/value pairs.

Syntax

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.toRecord({
    type : record.Type.TASK,
    id : taskRecordId,
    parameters: {'custparam_test':'helloWorld'}
});
...
```

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.</p> <p>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().</p> <p>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.</p>

Syntax

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.toSavedSearch({id: 234});
...
```

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.</p> <p>The internal ID is available only when the search is either loaded with <code>search.load(options)</code> or after it has been saved with <code>Search.save()</code>.</p> <p>Typical values are 55 or 234 or 87, not a value like <code>customsearch_mysearch</code>. Any ID prefixed with <code>customsearch</code> is a script ID, not the internal system ID for a search.</p>

Syntax

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.toSavedSearchResult({id: 234});
...

```

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	search.Search	required	

Syntax

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

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

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

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({
   scriptId: 31,
    deploymentId: 1,
    parameters: {'custparam_test':'helloWorld'}
});
...
```

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

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.id	string	required	The taskId for a tasklink For a list of supported task IDs, see the help topic Supported Tasklinks .

Parameter	Type	Required / Optional	Description
options.parameters	Object	optional	Contains additional URL parameters as key/value pairs.

Syntax

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.toTaskLink({
        id: ADMI_SHIPPING ,
        parameters: {'custparam_test':'helloWorld'}
    });
...

```

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 object containing a bill of materials.
	render.create()	render.TemplateRenderer	Creates a <code>render.TemplateRenderer</code> object
	render.mergeEmail(options)	render.EmailMergeResult	Creates a <code>render.EmailMergeResult</code> object
	render.packingSlip(options)	file.File	Creates a PDF or HTML containing a packing slip.

Member Type	Name	Return Type / Value Type	Description
	render.pickingTicket(options)	file.File	Creates a PDF or HTML containing a picking ticket.
	render.statement(options)	file.File	Creates a PDF or HTML containing a statement.
	render.transaction(options)	file.File	Creates a PDF or HTML 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.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
	TemplateRenderer.addRecord(options)	void	
	TemplateRenderer.addSearchResults(options)	void	
	TemplateRenderer.renderAsPdf()	Object	
	TemplateRenderer.renderAsPdfToResponse()	void	
	TemplateRenderer.renderAsString()	string	
	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	

Member Type	Name	Return Type / Value Type	Description
Property	TemplateRenderer.templateContent	string	

N/render Module Script Sample

Note: These sample scripts use the require function so that you can copy it into the debugger and test it. Keep in mind that you must use the define function in your entry point script (the script you attach to a script record). For additional information, see [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 renderTransactionToHtml() {
            var transactionFile = render.transaction({
                entityId: 23,
                printMode: render.PrintMode.HTML
            });
        }
        renderTransactionToHtml();
});
```

The following example renders a transaction record into a HTML page.

```
/** 
 * @NApiVersion 2.x
 */
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(record.Type.INVOICE, record.load({
                type: record.Type.INVOICE,
                id: 37
            }));
            var invoicePdf = renderer.renderAsPdf();
        }
        renderRecordToPdfWithTemplate();
});
```

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
*/
require(['N/render', 'N/file', 'N/record'],
    function(render, file, record) {
        function renderRecordToPdfWithTemplate() {
            var xmlTemplateFile = file.create({
                name: 'invoicePDFTemplate.xml',
                fileType: file.Type.XMLDOC,
                contents: '<div><!-- insert body information here --></div>'
            });
            var renderer = render.create();
            renderer.templateContent = xmlTemplateFile.getContents();
            renderer.addRecord(record.Type.INVOICE, record.create({
                type: record.Type.INVOICE,
            }));
            var invoicePdf = renderer.renderAsPdf();
        }
        renderRecordToPdfWithTemplate();
    });
});
```

render.EmailMergeResult

Object Description	Encapsulates an email merge result. Use render.mergeEmail(options) to create and return this object.
Supported Script Types	Server-side scripts
Module	N/render 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/render Module Script Sample](#).

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

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](#).

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

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](#).

render.TemplateRenderer

Object Description	Encapsulates a template engine object that produces HTML and PDF printed forms utilizing advanced PDF/HTML template capabilities. This object is available when the Advanced PDF/HTML Templates feature is enabled.
Supported Script Types	Server-side scripts
Module	N/render 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/render Module Script Sample](#).

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

Parameter	Type	Required / Optional	Description
options.format	render.DataSource	required	Data format
options.data	Object Document string	required	Object, document, or string

Syntax

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](#).

```
...
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();
...
```

TemplateRenderer.addRecord(options)

Method Description	Adds the record to a 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.record	record.Record object	required	The record to add

Syntax

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](#).

TemplateRenderer.addSearchResults(options)

Method Description	Adds search results to a 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

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](#).

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

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](#).

```
...  
var invoicePdf = renderer.renderAsPdf();  
...
```

TemplateRenderer.renderAsPdfToResponse()

Method Description	
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
http.ServerResponse	http.ServerResponse	required	

Syntax

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](#).

```
...  
var invoicePdf = renderer.renderAsPdfToResponse();  
...
```

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

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](#).

```
...  
var invoicePdf = renderer.renderAsString();  
...
```

TemplateRenderer.renderToResponse(options)

Method Description	Writes to an HTTP response object
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	

Syntax

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](#).

```
...  
var invoicePdf = renderer.renderToResponse();  
...
```

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

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](#).

```
...
var renderer = render.create();
renderer.setTemplateById(3);
var xml = renderer.renderAsString();
...
```

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

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](#).

```
...
var renderer = render.create();
renderer.setTemplateByScriptId("STDTMPLPRICELIST");
var xml = renderer.renderAsString();
...
```

TemplateRenderer.templateContent

Property Description	Content of template
Type	string
Module	N/render 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/render Module Script Sample](#).

```
...
renderer.templateContent = xmlTemplateFile.getContents();
...
```

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
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

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](#).

```
...
var transactionFile = render.bom({
    entityId : 23,
    printMode : render.PrintMode.HTML
});
...
```

render.create()

Method Description	Creates <code>render.TemplateRenderer</code> . 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 <code>render.xmlToPdf(options)</code> to produce a PDF. Note: To use this method, the Advanced PDF/HTML Templates feature must be enabled.
Returns	<code>render.TemplateRenderer</code>
Supported Script Types	Server-side scripts
Governance	None
Module	N/render 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/render Module Script Sample](#).

```
...
var renderer = render.create();
...
```

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
options.templateId	number	required	Internal ID of the template
options.entity	RecordRef	required	Entity
options.recipient	RecordRef	required	Recipient
options.customRecord	RecordRef	required	Custom record
options.supportCaseId	number	required	Support case ID
options.transactionId	number	required	Transaction ID

RecordRef

You can use a RecordRef to designate the record to perform the mail merge on.

Note: The RecordRef object encapsulates the type and ID of a particular record instance.

Property	Type	Required / Optional	Description
RecordRef.id	number	required	Internal ID of the record instance
RecordRef.type	string	required	The record type ID

Syntax

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](#).

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

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.entityId	number	required	The internal ID of the packing slip to print
options.printMode	string	optional	The print output type. Set using the render.PrintMode enum. By default, uses the company/user preference for print output.
options.formId	number	optional	The packing slip form number
options.fulfillmentId	number	optional	Fulfillment ID number

Syntax

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](#).

```
...
var transactionFile = render.packingSlip({
    entityId: 23,
    printMode: render.PrintMode.HTML
});
...
```

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

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.
options.formId	number	optional	The packing slip form number
options.shipgroup	number	optional	Shipping group for the ticket
options.location	number	optional	Location for the ticket

Syntax

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](#).

```
...
var transactionFile = render.pickingTicket({
    entityId: 23,
    printMode: render.PrintMode.HTML
});
...
...
```

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.

Parameter	Type	Required / Optional	Description
options.formId	number	optional	Internal ID of the form to use to print the statement
options.startDate	Date	optional	Date of the oldest transaction to appear on the statement
options.statementDate	Date	optional	Statement date
options.TransactionsOnly	boolean true false	optional	Include only open transactions

Syntax

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](#).

```
...
var transactionFile = render.statement({
    entityId: 23,
    printMode: render.PrintMode.HTML
});
...
...
```

render.transaction(options)

Method Description	Use this method to create a PDF or HTML object of a transaction. Note: File size is limited to 5MB. 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 <i>Advanced PDF/HTML Templates</i>
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

Parameter	Type	Required / Optional	Description
options.printMode	enum	optional	The print output type. Set using the <code>render.PrintMode</code> enum. By default, uses the company/user preference for print output.
options.formId	number	optional	The transaction form number

Syntax

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](#).

```
...
var transactionFile = render.transaction({
    entityId: 23,
    printMode: render.PrintMode.HTML
});
...
...
```

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 5MB.
Returns	<code>file.File</code>
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	<code>xml.Document</code> string	required	Entire XML document or string to convert to PDF

Syntax

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](#).

```
...  
var pdfFile = render.xmlToPdf({  
    xmlString: xmlStr  
});  
...
```

render.PrintMode

Enum Description	Holds the string values for supported print output types. Use this enum to set the options.printMode 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

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](#).

```
...  
printMode: render.PrintMode.HTML  
...
```

N/runtime Module

Load the runtime module when you want to access the current runtime settings for the script and script deployment, the user currently executing the script, and user-defined sessions.

- [N/runtime Module Members](#)
- [Script Object Members](#)
- [Session Object Members](#)
- [User Object Members](#)
- [N/runtime Module Script Sample](#)

N/runtime Module Members

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.
	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. These are the features that appear on the Enable Features page at Setup > Company > Enable Features.
Property	runtime.accountId	string (read-only)	Returns the account ID for the currently logged-in user.
	runtime.envType	runtime.EnvType	Returns a runtime.EnvType enumeration that represents the environment in which the current 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 runtime.executionContext property of the N/runtime Module .

Member Type	Name	Return Type / Value Type	Description
	runtime.EnvType	enum	Enumeration that holds the type of environment for the currently running script. Returned by the <code>runtime.envType</code> property of the N/runtime Module.
	runtime.Permission	enum	Enumeration that holds the user permission level for a specific permission ID. Returned by the <code>User.getPermission(options)</code> method.

Script Object Members

Member Type	Name	Return Type / Value Type	Description
Method	<code>Script.getParameter(options)</code>	number Date string boolean	Returns the value of a script parameter for the currently executing script.
	<code>Script.getRemainingUsage()</code>	number	Returns a number value for the usage units remaining for the currently executing script.
Property	<code>Script.deploymentId</code>	string (read-only)	Returns the deployment ID for the script deployment on the currently executing script.
	<code>Script.id</code>	string (read-only)	Returns the script ID for the currently executing script.
	<code>Script.logLevel</code>	string (read-only)	Returns the script logging level for the current script execution. This method is not supported on client scripts.
	<code>Script.percentComplete</code>	number (read-only)	Return the percent complete specified for the current scheduled script execution. The return value will appear in the % Complete column in the Scheduled Script Status page.
	<code>Script.bundleIds</code>	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	<code>Session.get(options)</code>	string	Returns the user-defined session object value associated with the session object key.
	<code>Session.set(options)</code>	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 <code>runtime.Permission</code> enumeration.
	User.getPreference(options)	string	Returns the value of a NetSuite preference. Currently only General Preferences and Accounting Preferences are exposed in SuiteScript.
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

The following Suitelet sample writes user and session information for the currently executing script to the response.

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/runtime'],
    function(runtime) {
        function onRequest(context) {
            var remainingUsage = runtime.getCurrentScript().getRemainingUsage();
            var userRole = runtime.getCurrentUser().role;
            runtime.getCurrentSession().set('scope', 'global');
            var sessionScope = runtime.getCurrentSession().get('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
        };
});
});
```

runtim.e.Script

Object Description	Encapsulates the runtime settings of the currently executing script. Use runtime.getCurrentScript() to return this object.
Supported Script Types	Server-side scripts
Module	N/runtime 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/runtime Module Script Sample](#).

```
...
var scriptObj = runtime.getCurrentScript();
log.debug('Script ID: ' + scriptObj.id);
...;
```

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	<ul style="list-style-type: none"> The name of the script parameter. 	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/runtime Module Script Sample](#).

```
...
var scriptObj = runtime.getCurrentScript();
log.debug("Script parameter of custscript1: " +
    scriptObj.getParameter({name: 'custscript1'}));
...
```

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

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](#).

```
...
var scriptObj = runtime.getCurrentScript();
log.debug("Remaining governance units: " + scriptObj.getRemainingUsage());
...
```

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

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](#).

```
...
var scriptObj = runtime.getCurrentScript();
log.debug("Deployment Id: " + scriptObj.deploymentId);
...
```

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

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](#).

```
...
var scriptObj = runtime.getCurrentScript();
log.debug("Script Id: " + scriptObj.scriptId);
...
```

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: <ul style="list-style-type: none">• DEBUG• AUDIT• ERROR• EMERGENCY
Type	string (read-only)
Module	N/runtime 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/runtime Module Script Sample](#).

```
...
var scriptObj = runtime.getCurrentScript();
log.debug("Logging level: " + scriptObj.logLevel);
...
```

Script.percentComplete

Property Description	Return the percent complete specified for the current scheduled script execution. The return value will appear in the % Complete column in the Scheduled Script Status page.
Important:	This property throws SSS_OPERATION_UNAVAILABLE if the currently executing script is not a scheduled script.
Type	number (read-only)
Module	N/runtime 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/runtime Module Script Sample](#).

```
...
var scriptObj = runtime.getCurrentScript();
if (scriptObj.executionContext == ContextType.SCHEDULED)
{
    log.debug("Script percent complete: " + scriptObj.percentComplete);
    ...
}
...
```

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

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](#).

```
...
var scriptObj = runtime.getCurrentScript();
var bundleArr = scriptObj.bundleIds;
```

...

runtime.Session

Object Description	Encapsulates the user session for the currently executing script. Use this object to set and get user-defined objects for the current user session. Use the objects to track user-related session data. For example, you can gather information about the user scope, budget, or business problems. Use Session.set(options) to set session object values and then use Session.get(options) to retrieve the values.
Supported Script Types	Server-side scripts
Module	N/runtime 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/runtime Module Script Sample](#).

```
...
var sessionObj = runtime.getCurrentSession();
sessionObj.set({name: "myKey", value: "myValue"});
log.debug("Session object myKey value: " + sessionObj.get({name: "myKey"}));
...
```

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 runtime.Session .	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 link back to [N/runtime Module Script Sample](#).

```
...
var sessionObj = runtime.getCurrentSession();
sessionObj.set({name: "myKey", value: "myValue"});
log.debug("Session object myKey value: " + sessionObj.get({name: "myKey"}));
...
```

Session.set(options)

Method Description	Sets a key and value for a user-defined <code>runtime.Session</code> . Use Session.get(options) 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

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](#).

```
...
var sessionObj = runtime.getCurrentSession();
sessionObj.set({
    name: "myKey",
    value: "myValue"
});
log.debug("Session object myKey value: " + sessionObj.get({name: "myKey"}));
...
```

runtime.User

Object Description	Encapsulates the properties and preferences for the user of the currently executing script.
Supported Script Types	Server-side scripts
Module	N/runtime 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/runtime Module Script Sample](#).

```
...
var userObj = runtime.getCurrentUser();
...
```

User.getPermission(options)

Method Description	Returns a user permission level for the specified permission as a runtime.Permission enumeration.
Returns	string
Supported Script Types	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 Permission Names and IDs .	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 link back to [N/runtime Module Script Sample](#).

```
...
var userObj = runtime.getCurrentUser();
log.debug("User permission of ADMI_ACCOUNTING: " +
```

```
(userObj.getPermission('ADMI_ACCOUNTING') ==
    runtime.Permission.FULL?'FULL':userObj.getPermission('ADMI_ACCOUNTIN
G'));
...

```

User.getPreference(options)

Method Description	Returns the value of a NetSuite preference. Currently only General Preferences and Accounting Preferences are exposed in SuiteScript. You can view General Preferences by going to Setup > Company > General Preferences. View Accounting Preferences by going to Setup > Accounting > Accounting Preferences. If you want to change the value of a General or Accounting preference using SuiteScript 2.0, you must load each preference page using config.load(options) , where options.name is COMPANY_PREFERENCES or ACCOUNTING_PREFERENCES. The config.load(options) method returns a record.Record . You can use the Record.setValue(fieldName, value) method to set the preference. Note: The permission level will be Permission.FULL if the script is configured to execute as admin. You can configure a script to execute as admin by selecting "administrator" from the Execute as Role field on Script Deployment page.
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 the preference. For a list of preference IDs, see Preference Names and IDs .	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 link back to [N/runtime Module Script Sample](#).

```
...
var userObj = runtime.getCurrentUser();
log.debug("User preference for emailemployeeonapproval: " + userObj.getPreference({name: "email
employeeonapproval"}));
```

...

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

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](#).

```
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user department: " + userObj.department);
...
```

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

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](#).

```
...
var userObj = runtime.getCurrentUser();
log.debug("Current user email: " + userObj.email);
...
```

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

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](#).

```
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user: " + userObj.id);
...
```

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

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](#).

```
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user location: " + userObj.location);
...
```

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

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](#).

```
...
```

```
var userObj = runtime.getCurrentUser();
log.debug("Name of current user: " + userObj.name);
...
```

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

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](#).

```
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user role: " + userObj.role);
...
```

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

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](#).

```
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user center type (role center): " + userObj.roleCenter);
...
```

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

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](#).

```
...
var userObj = runtime.getCurrentUser();
log.debug("Custom script ID of current user role: " + userObj.roleId);
...
```

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

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](#).

```
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user subsidiary: " + userObj.subsidiary);
...
```

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

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](#).

```
...  
var scriptObj = runtime.getCurrentScript();  
...
```

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	runtime.Session
Supported Script Types	Server-side scripts
Governance	None
Module	N/runtime 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/runtime Module Script Sample](#).

```
...  
var sessionObj = runtime.getCurrentSession();  
...
```

runtime.getCurrentUser()

Method Description	Returns a runtime.User that represents the properties and preferences for the user of the currently executing script. Use this method to get session objects for the current user session. If you want to get properties for the script or session, use runtime.getCurrentScript() or runtime.getCurrentSession() instead.
---------------------------	---

Returns	runtime.User
Supported Script Types	Server-side scripts
Governance	None
Module	N/runtime 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/runtime Module Script Sample](#).

```
...
var userObj = runtime.getCurrentUser();
...
```

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.name	string	Required	The name of the parameter to check.	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/runtime Module Script Sample](#).

```
...
log.debug('Advanced Billing feature is enabled: ' + runtime.isFeatureInEffect({name: "ADVBILLING"}));
```

...

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

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](#).

```
...
log.debug("Account ID for the current user: " + runtime.accountId);
...
```

runtime.envType

Property Description	Returns a runtime.EnvType enumeration that represents the environment in which the current script is executing.
Type	runtime.EnvType
Module	N/runtime 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/runtime Module Script Sample](#).

```
...
log.debug("Environment for current user: " + JSON.stringify(runtime.envType));
...
```

runtime.executionContext

Property Description	Returns a runtime.ContextType enumeration that represents what triggered the current script.
Type	runtime.ContextType
Module	N/runtime 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/runtime Module Script Sample](#).

```
...
log.debug("Execution context for current script: " + JSON.stringify(runtime.executionContext));
...
```

runtime.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

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](#).

```
...
log.debug("Number of queues available: " + runtime.queueCount);
...
```

runtime.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

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](#).

```
...
log.debug("Current NetSuite version: " + runtime.version);
...
```

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 . 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/runtime Module
Since	Version 2015 Release 2

Values

- | | | |
|--|---|---|
| <ul style="list-style-type: none"> • CSV_IMPORT • CUSTOM_MASSUPDATE • SCHEDULED | <ul style="list-style-type: none"> • SUITELET • USEREVENT • USER_INTERFACE | <ul style="list-style-type: none"> • WEBSERVICES • WEBSTORE • WORKFLOW |
|--|---|---|

runtime.EnvType

Enum Description	Enumeration that holds the type of environment for the currently running script. Returned by the <code>runtime.envType</code> property of the N/runtime 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/runtime Module
Since	Version 2015 Release 2

Values

- | |
|---|
| <ul style="list-style-type: none"> • SANDBOX • PRODUCTION • BETA • INTERNAL |
|---|

runtime.Permission

Enum Description	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 Understanding NetSuite Permissions . 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/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.
	search.create.promise(options)	search.Search	Creates a new search asynchronously and returns it as a <code>search.Search</code> object.
	search.load(options)	search.Search	Loads an existing saved search and returns it as a <code>search.Search</code> object.
	search.load.promise(options)	search.Search	Loads an existing saved search asynchronously and returns it as a <code>search.Search</code> object.
	search.delete(options)	void	Deletes an existing saved search asynchronously and returns it as a <code>search.Search</code> object.
	search.delete.promise(options)	void	Deletes an existing saved search and returns it as a <code>search.Search</code> object.

Member Type	Name	Return Type / Value Type	Description
	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	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	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.
	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.
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 customsearch.
	Search.isPublic	boolean true false	Value is true if the search is public, or false 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.
	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

Member Type	Name	Return Type / Value Type	Description
	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.
	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 <code>Search.runPaged(options)</code> 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 <code>Search.runPaged(options)</code> was executed.

PageRange Object Members

The following members are called on `search.PageRange`.

Member Type	Name	Return Type / Value Type	Description
	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.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.each(callback)	void	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 search.Column objects that represent the columns returned in the search results.

N/search Module Script Samples

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 creates a saved search on the sales order record.

```
/**  
 * @NApiVersion 2.x  
 */  
require(['N/search'],  
    function(search) {  
        function createSearch() {  
            var mySalesOrderSearch = search.create({  
                type: 'salesorder',  
                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();  
    });
```

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('entity');  
                var subsidiary = result.getValue('subsidiary');  
                return true;  
            });  
        }  
        loadAndRunSearch();  
    });
```

The following example creates 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(0, 100);  
            for (var i = 0; i < searchResult.length; i++) {  
                var entity = searchResult[i].getValue('entity');  
                var subsidiary = searchResult[i].getValue('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('entity');  
                    var subsidiary = result.getValue('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.</p>
Supported Script Types	All script types
Module	N/search 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/search Module Script Samples](#).

```
...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});
...
```

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

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](#).

```
...
function loadAndRunSearch() {
    var mySearch = search.load({
        id: 'customsearch_my_so_search'
    });
    mySearch.run().each(function(result) {
        var entity = result.getValue('entity');
        var subsidiary = result.getValue('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

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.pageSize	number	optional	<p>Maximum number of entries per page</p> <p>There is an upper limit, a lower limit, and a default setting:</p> <ul style="list-style-type: none"> 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

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](#).

```
...
var pagedData = mySearch.runPaged({pageSize:50});
```

...

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

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](#).

```
...
mySearch.runPaged.promise().then(getPageRangesPromiseChain);
...
```

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 . Note: You do not need to set these properties if you load a previously saved search with search.load(options) and then save it. This method also includes a promise version, Search.save.promise() . For more information about promises, see Promise object .
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 <code>Search.title</code> property not set on <code>search.Search</code> .
NAME_ALREADY_IN_USE	A search has already been saved with that name. Please use a different name.	The <code>Search.title</code> property on <code>search.Search</code> is not unique.
SSS_DUPLICATE_SEARCH_SCRIPT_ID		The <code>Search.id</code> property on <code>search.Search</code> is not unique.

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](#).

```
...
mySalesOrderSearch.save();
...
```

Search.save.promise()

Method Description	Asynchronously saves a search created by <code>search.create(options)</code> or loaded with <code>search.load(options)</code> . Returns the internal ID of the saved search. Note: For more information about using this method, see Search.save() . For additional information on promises, see Promise object .
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

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](#).

```
...
```

```

search.create.promise({
    type: 'salesorder'
})
.then(function(searchObj) {
    return searchObj.save.promise()
})
.then(function (result) {
    log.debug("Completed: " + result);
    // do something after completion
})
.catch(function onRejected(reason) {
    // do something on rejection
});
...

```

Search.searchType

Property Description	Internal ID name of the record type on which a search is based. Use this if you have the internal ID of the search, but do not know the record type the search was based on. For example, if the search was on a Customer record, this property is customer; if the search was on the Sales Order record type, this property is salesorder.
Type	read-only string
Module	N/search 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/search Module Script Samples](#).

```

...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});
log.debug('record type: ' + mySearch.searchType);
...

```

Search.searchId

Property Description	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 customsearch_myssearch. 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

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](#).

```
...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});
log.debug("search id #: " + mySearch.searchId);
...
```

Search.filters

Property Description	Filters for the search as an array of search.Filter objects. Value is <code>null</code> if the search has no defined filters. You set this value with an array or single search.Filter objects to overwrite any prior filters. Use <code>null</code> to set an empty array and remove any existing filters on this search. Use search.createFilter(options) to create a filter. Note: If you want to get or set a search filter expression, use the Search.filterExpression property.
Type	search.Filter[]
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

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](#).

```
...
var myFilter = search.createFilter({
    name: 'entity',
    operator: 'ISEMPTY',
});

function createSearch() {
    var mySalesOrderSearch = search.create({
        type: 'salesorder',
        filters: myFilter
});
```

...

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 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. Note: If you want to get or set a search filters, use the <code>Search.filters</code> property. For more information, see <i>Search Filter Expression Overview</i> .
Type	Object[]
Module	N/search Module
Since	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_INVALID_SRCH_FILTER_EXPR		An invalid search filter expression was set

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](#).

```
...
function createSearch() {
    var mySalesOrderSearch = search.create({
        type: 'salesorder',
        filters: [
            ['mainline', 'is', 'T'],
            'and', ['subsidiary.name', 'contains', 'CAD']
        ]
    });
...
}
```

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 `null` to set an empty array and remove any existing columns on this search.

Type	search.Column[] string[]
Module	N/search Module
Since	Version 2015 Release 2

Errors

Error Code	Message	Thrown If
SSS_INVALID_SRCH_COLUMN		The value passed in was not a string or search.Column Object

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](#).

```
...
function createSearch() {
    var mySalesOrderSearch = search.create({
        type: 'salesorder',
        columns: ['entity', 'subsidiary', 'name', 'currency'],
    });
...
}
```

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 search.create(options) . The Search.title property is required to save a search with Search.save() .
Type	string
Module	N/search 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/search Module Script Samples](#).

```
...
function createSearch() {
    var mySalesOrderSearch = search.create({
        type: 'salesorder',
        title: 'My SalesOrder Search',
    });
}
```

```

        id: 'customsearch_my_so_search',
    });
mySalesOrderSearch.save();
...

```

Search.id

Property Description	Script ID for a saved search, starting with customsearch. If you do not set this property and then save the search, NetSuite generates a script ID for you.
	Note: This is not the internal NetSuite ID for the saved search. See Search.searchId.
Type	number
Module	N/search 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/search Module Script Samples](#).

```

...
function createSearch() {
    var mySalesOrderSearch = search.create({
        type: 'salesorder',
        title: 'My SalesOrder Search',
        id: 'customsearch_my_so_search',
    });
...

```

Search.isPublic

Property Description	Value is true if the search is public, or false if it is not. By default, all searches created through search.create(options) are private.
Type	boolean true false
Module	N/search 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/search Module Script Samples](#).

```

...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});
mySearch.isPublic = true;

```

...

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.
	<p>Note: Use <code>search.ResultSet</code> for the set of results from a search.</p> <p>For more information about executing NetSuite searches using SuiteScript, see Searching Overview.</p>
Supported Script Types	All script types
Module	N/search 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/search Module Script Samples](#).

```
...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});

var searchResult = mySearch.run().getRange(0, 100);
for (var i = 0; i < searchResult.length; i++) {
    var entity = searchResult[i].getValue({
        name: 'entity'
    });
    var subsidiary = searchResult[i].getValue({
        name: 'subsidiary'
    });
}
...
```

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>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.
options.summary	search.Summary	Optional	The summary type for this column. See search.Summary .

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](#).

```
...
var searchResult = mySearch.run().getRange(0, 100);
for (var i = 0; i < searchResult.length; i++) {
    var entity = searchResult[i].getValue({
        name: 'entity'
    });
...
}
```

Result.getText(options)

Method Description	The UI display name, or text value, for a search result column. Note: The following field types are supported: select, image, or document fields.
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.

Parameter	Type	Required / Optional	Description
options.summary	search.Summary	Optional	The summary type used for the search column. See search.Summary .

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](#).

```
...
var searchResult = mySearch.run().getRange(0, 100);
for (var i = 0; i < searchResult.length; i++) {
    var entity = searchResult[i].getText({
        name: 'entity'
    });
...
}
```

Result.recordType

Property Description	The type of record returned in a search result row.
Type	string (read-only)
Module	N/search Module
Since	Version 2015 Release 2

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

Result.columns

Property Description	Array of search.Column objects that encapsulate the columns returned in the search result row.
Type	search.Column[]
Module	N/search 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/search Module Script Samples](#).

search.Column

Object Description	Encapsulates a single search column in a <code>search.Search</code> . Use the methods and properties available to the Column object to get or set Column properties. You create a search column object with <code>search.createColumn(options)</code> and add it to a <code>search.Search</code> object that you create with <code>search.create(options)</code> or load with <code>search.load(options)</code> . In addition, <code>search.ResultSet</code> contains an array of Column objects returned in the results of a search.
Supported Script Types	All script types
Module	N/search Module
Since	Version 2015 Release 2

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. Note: You can only use this method if you use MIN or MAX as the summary type on a search column with the <code>Result.getValue(options)</code> method.
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.

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

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](#).

```
...
log.debug({
    details: 'Search Column Name: '
        + columnObj.name
});
...
```

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

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](#).

```
...
log.debug({
    details: 'Join ID for Search Column: '
        + columnObj.join
});
...
```

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

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](#).

```
...
```

```
log.debug({
    details: 'Summary Type for Search Column: '
        + columnObj.summary
});
...

```

Column.formula

Property Description	Formula used for a search column as a string. 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: {firstname} " , " {lastname} In the above formula, <code>firstname</code> and <code>lastname</code> are script IDs for the fields on the Customer record form. To set this value, you must use <code>formulatext</code> , <code>formulanumeric</code> , <code>formuladatetime</code> , <code>formulapercent</code> , or <code>formulacurrency</code> . For more information, see <i>Using Formulas, Special Functions, and Sorting in Search</i> .
Type	string
Module	N/search Module
Since	Version 2015 Release 2

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

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](#).

```
...
var columnObj = search.createColumn({
    name: 'formulanumeric'
    label: 'Numeric Formula'
});
...

```

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	
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		Unknown function is set.

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](#).

```
...
var columnObj = search.createColumn({
```

```

        name: 'customer'
        function: 'ageInYears'
    });
...

```

Column.sort

Property Description	The sort order of the column. Use the search.Sort enum to set the value. If <code>Column.sort</code> is not set, the column is not sorted in any particular order.
Type	<code>search.Sort</code> enum
Module	N/search 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/search Module Script Samples](#).

```

...
var columnObj = search.createColumn({
    name: 'invoice'
    function: 'ASC'
});
...

```

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 <code>search.createFilter(options)</code> and add it to a <code>search.Search</code> object that you create with <code>search.create(options)</code> or load with <code>search.load(options)</code> . Note: NetSuite uses an implicit AND operator with search filters, as opposed to filter expressions which explicitly use either AND and OR operators. Use the following guidelines with the Filter object: <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 <code>Filter.formula</code> 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.
Supported Script Types	All script types
Module	N/search 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/search Module Script Samples](#).

```
...
var mySearchFilter = search.createFilter({
    name: 'entity',
    operator: 'ISEMPTY',
});
...
```

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

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](#).

```
...
log.debug({
    details: 'Filter Name: '
        + filterObj.name
});
...
```

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

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](#).

```
...
log.debug({
    details: 'Join ID: '
        + filterObj.join
});
...

```

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

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](#).

```
...
log.debug({
    details: 'Operator Used: '
        + filterObj.operator
});
...

```

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		Unknown summary type is set.

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](#).

```
...
```

```
var mySearchFilter = search.createFilter({  
    name: 'entity',  
    summary: 'GROUP',  
});  
...
```

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

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](#).

```
...  
log.debug({  
    details: 'Search Filter Formula: '  
        + filterObj.formula  
});  
...
```

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.
Supported Script Types	All script types
Module	N/search 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/search Module Script Samples](#).

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

```

ResultSet.getRange(options)

Method Description	Retrieve a slice of the search result as an array of <code>search.Result</code> 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, <code>getRange(0, 10)</code> 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 <code>getRange(20, 30)</code> will return an array of 5 <code>search.Result</code> objects.
Returns	<code>search.Result[]</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
<code>options.start</code>	number	Required	Index number of the first result to return, inclusive.
<code>options.end</code>	number	Required	Index number of the last result to return, exclusive.

Syntax

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](#).

```
...
```

```
var results = rs.getRange({
    start: 0,
    end: 1000
});
...
```

ResultSet.each(callback)

Method Description	<p>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:</p> <pre>boolean callback(result.Result result);</pre> <p>The callback function takes a search.Result object as an input parameter and returns a boolean which can be used to stop the iteration with a value of <code>false</code>, or continue the iteration with a value of <code>true</code>.</p> <p>Important: The work done in the context of the callback function counts towards the governance of the script that called it. For example, if the callback function is running in the context of a scheduled script, which has a 10,000 unit governance limit, make sure the amount of processing within the callback function does not put the entire script at risk of exceeding scheduled script governance limits.</p>
Returns	void
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 search.Result object.

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](#).

```
...
mySearch.run().each(function(result) {
    var entity = result.getValue({
        name: 'entity'
```

```

    });
    var subsidiary = result.getValue({
        name: 'subsidiary'
    });
    return true;
});
...

```

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

search.Page

Object Description	Encapsulates an individual search page containing a result set for a paginated search.
Supported Script Types	All script types
Module	N/search Module
Since	Version 2015 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/search Module Script Samples](#).

```

...
var page = pagedData.fetch({index: lastPageRange.index});
...

```

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

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](#).

```
...
while (!page.isFirst){
    page = page.next();
}
...
```

Page.next.promise()

Method Description	Method used to asynchronously fetch the next segment of data (bounded by search.PageRange). Moves the current page to another range. The promise is complete when the data for this range is loaded or rejected. Note: For information about errors thrown for this method, see Page.next() . For additional information on promises, see Promise object .
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

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](#).

```
...
return mypage.next.promise().then(processPage);
...
```

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

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](#).

```
...
while (!page.isLast){
    page = page.prev();
}
...
```

Page.prev.promise()

Method Description	Method used to asynchronously fetch the previous segment of data (bounded by search.PageRange). Moves the current page to another range. The promise is complete when the data for this range is loaded or rejected. Note: For information about errors thrown for this method, see Page.prev() . For additional information on promises, see Promise object .
Returns	Void
Synchronous Version	Page.prev()
Supported Script Types	All client-side scripts
Governance	5 units
Module	N/search 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 promise script example, see [Promise object](#).

```
...
return mypage.prev.promise().then(processPage);
...
```

Page.data

Property Description	The results from a paginated search.
Type	search.Result[] This property is read-only.
Module	N/search 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/search Module Script Samples](#).

```
...
function processPage(page){
    page.data.forEach(function(value){
        log.debug("data: " + page.data);
    });
...
}
```

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

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](#).

```
...
while (!page.isFirst){
```

```
page = page.next();
...
```

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

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](#).

```
...
while (!page.isLast){
    page = page.prev();
...
}
```

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

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](#).

```
...
var lastPageRange = pagedData.pageRanges[pagedData.pageRanges.length - 1];
...
```

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

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](#).

```
...
log.debug("Page Range: " + mySearchPage.pageRange);
...
```

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.
Supported Script Types	All script types
Module	N/search Module
Since	Version 2015 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/search Module Script Samples](#).

```
...
var pagedData = mySearch.runPaged({pageSize:1000});
...
```

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	Void
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

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](#).

```
...
var page = pagedData.fetch({index: lastPageRange.index});
...
```

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	Void
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

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](#).

```
...  
return pagedData.fetch.promise().then(processPage);  
...
```

PagedData.count

Property Description	The total number of results when Search.runPaged(options) was executed.
Type	number This property is read-only.
Module	N/search 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/search Module Script Samples](#).

```
...  
log.debug("Result Count: " + myPagedData.count);  
...
```

PagedData.pageRanges

Property Description	The collection of PageRange objects that divide the entire result set into smaller groups. Includes page range information with the key and label for rendering.
Type	search.PageRange[] This property is read-only.
Module	N/search 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/search Module Script Samples](#).

```
...  
log.debug("PageRange Array: " + myPagedData.pageRanges);  
...
```

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

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](#).

```
...
log.debug("Max Page Size: " + myPagedData.pageSize);
...
```

PagedData.searchDefinition

Property Description	The search criteria used to execute the result set for this PagedData Object.
Type	read-only search.Search
Module	N/search 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/search Module Script Samples](#).

```
...
log.debug("Search Details: " + myPagedData.searchDefinition);
...
```

search.PageRange

Object Description	Defines the page range to contain the result set
Supported Script Types	All script types
Module	N/search Module
Since	Version 2015 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/search Module Script Samples](#).

```
...  
var page = pagedData.fetch(lastPageRange);  
...
```

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

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](#).

```
...  
log.debug("Page Range Description: " + myPageRange.compoundLabel);  
...
```

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

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](#).

```
...  
log.debug("Page Range Index: " + myPageRange.index);  
...
```

search.create(options)

Method Description	Creates a new search and returns it as a search.Search object. The search can be modified and run as an ad-hoc search with Search.run() , without saving it. Alternatively, calling Search.save() will save the search to the database, so it can be reused later in the UI or loaded with search.load(options) .
---------------------------	--

	<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 search.Type enum for this argument.	Version 2015 Release 2
<code>options.filters</code>	<code>search.Filter[] Object[]</code>	Optional	A single <code>search.Filter</code> object, an array of <code>search.Filter</code> objects, or a search filter expression. Note: You can further filter the returned <code>search.Search</code> object by adding additional filters with Search.filters or Search.filterExpression .	Version 2015 Release 2
<code>options.columns</code>	<code>search.Column[] string[]</code>	Optional	A single <code>search.Column</code> object or array of <code>search.Column</code> objects.	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 Search.save() .	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 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.
SSS_INVALID_SRCH_FILTER_EXPR	Malformed search filter expression. This is a general error raised when a filter expression cannot be parsed. For example: [f1, 'and', 'and', f2]	The options.filters parameter is not a valid search filter, filter array, or filter expression.
SSS_INVALID_SRCH_COLUMN	An search.Column Object contains an invalid column, or is not in proper syntax: {1}.	The options.columns parameter is not a valid column, string, or column or string array.

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](#).

```
...
function createSearch() {
    var mySalesOrderSearch = search.create({
        type: 'salesorder',
        title: 'My SalesOrder Search',
        id: 'customsearch_my_so_search',
        columns: ['entity', 'subsidiary', 'name', 'currency'],
        filters: [
            ['mainline', 'is', 'T'],
            'and', ['subsidiary.name', 'contains', 'CAD']
        ]
    });
...
}
```

search.create.promise(options)

Method Description	Creates a new search asynchronously and returns it as a search.Search object.
	Note: For information about the parameters and errors thrown for this method, see search.create(options) . For additional information on promises, see Promise object.
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

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](#).

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

```

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.

Error Code	Message	Thrown If
INVALID_SEARCH	That search or mass update does not exist.	Cannot find saved search with the saved search ID from options.id parameter.

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](#).

```
...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});
...
```

search.load.promise(options)

Method Description	Loads an existing saved search asynchronously and returns it as a search.Search object. The saved search could have been created using the UI or created with search.create(options) and Search.save() . Note: For information about the parameters and errors thrown for this method, see search.load(options) . For additional information on promises, see Promise object.
Returns	search.Search
Synchronous Version	search.load(options)
Supported Script Types	All client-side scripts
Governance	5 units
Module	N/search 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 promise script example, see [Promise object](#).

```
...
search.load.promise({
    type : 'salesorder',
    id : 'customsearch_txn_search_salesorder'
})
.then(function (result) {
    log.debug("Completed: " + result);
    // do something after completion
})
.catch(function onRejected(reason) {
```

```
// do something on rejection
});
...

```

search.delete(options)

Method Description	Deletes an existing saved search. The saved search could have been created using the UI or created with search.create(options) and Search.save() . The search.delete(options) method also includes a promise version, search.delete.promise(options) . For more information about promises, see Promise object .
Returns	void
Supported Script Types	All script types
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

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](#).

```
...
search.delete({
    id: 'customsearch_my_so_search'
});
...

```

search.delete.promise(options)

Method Description	Deletes an existing saved search asynchronously and returns it as a search.Search object. The saved search can be created using the UI or created with search.create(options) and Search.save() . Note: For information about the parameters and errors thrown for this method, see search.delete(options) . For additional information on promises, see Promise object .
Returns	<code>void</code>
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

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](#).

```
...
search.delete.promise({id: 'customsearch_txn_search_salesorder'})
    .then(function(){
        search.load({id: 'customsearch_txn_search_salesorder'});
    })
    .catch(function onRejected(reason) {
        log.debug('Invalid search: ' + reason.name);
    });
...
...
```

search.duplicates(options)

Method Description	Performs a search for duplicate records based on the account's duplicate detection configuration. The search.duplicates(options) method also includes a promise version, search.duplicates.promise(options) . For more information about promises, see Promise object . Important: This API is for only records that support duplicate record detection. For example, customers, leads, prospects, contacts, partners, and vendors records. For more information about duplicate record detection, see the help topic Duplicate Record Detection
---------------------------	--

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	string	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

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](#).

```
...
var duplicatesRecords = search.duplicates({
    type: 'CONTACTS',
```

```
    id: 'customsearch_my_contacts_search'
});
```

search.duplicates.promise(options)

Method Description	Performs a search for duplicate records asynchronously based on the Duplicate Detection configuration for the account. Returns an array of search.Result objects. This method only applies to records that support duplicate record detection. These records include customer lead prospect partner vendor contact.
Note:	For information about the parameters and errors thrown for this method, see search.duplicates(options) . For additional information on promises, see Promise object .
Returns	search.Result[]
Synchronous Version	search.duplicates(options)
Supported Script Types	All client-side scripts
Governance	10 units
Module	N/search 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 promise script example, see [Promise object](#).

```
...
search.duplicates.promise({
    type: 'customer',
    id: 28
})
.then(function (result) {
    log.debug("Completed: " + result);
    // do something after completion
})
.catch(function onRejected(reason) {
    // do something on rejection
});
...
```

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:
---------------------------	--

	<p>'cu: simpson'</p> <p>The search.global(options) method also includes a promise version, search.global.promise(options). For more information about promises, see Promise object.</p> <p>For more information about global search, see the help topic Global Search.</p>
Returns	<p>search.Result[] as an array of result objects containing these columns: name, type, info1, and info2</p> <p>Results are limited to 1000 records.</p> <p>If there are no search results, this method returns <code>null</code>.</p>
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.keywords	string	Required	Global search keywords string or expression.	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

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](#).

```
...
var customerSearch = search.global({
    keywords: 'cu: simpson'
});
```

search.global.promise(options)

Method Description	Performs a global search asynchronously against a single keyword or multiple keywords.
---------------------------	--

	Returns an array of <code>search.Result</code> objects with four columns: name , type , info1 , and info2 .
Note:	For information about the parameters and errors thrown for this method, see <code>search.global(options)</code> . For additional information on promises, see Promise object .
Returns	<code>search.Result[]</code>
Synchronous Version	<code>search.global(options)</code>
Supported Script Types	All client-side scripts
Governance	10 units
Module	N/search 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 promise script example, see [Promise object](#).

```
...
search.global.promise({
    keywords: 'Alan Rath'
})
.then(function (result) {
    log.debug("Completed: " + result);
    // do something after completion
})
.catch(function onRejected(reason) {
    // do something on rejection
});
...
...
```

search.lookupFields(options)

Method Description	Performs a search for one or more body fields on a record. For example, this method returns results in the following form:
	<pre>{ internalid: 1234, firstname: 'Joe', my_select: [{ value: 1, text: 'US Sub' }], my_multiselect: [{ value: 1, text: 'US Sub' }]}</pre>

```

    },{
      value: 2,
      text: 'EU Sub'
    }]
}

```

You can use joined-field lookups with this method, with the following syntax:

`join_id.field_name`

The `search.lookupFields(options)` method also includes a promise version, `search.lookupFields.promise(options)`. For more information about promises, see [Promise object](#).

Returns	Object Returns select fields as an object with value and text properties. Returns multiselect fields as an 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>	string	Required	Record internal ID name for which you want to look up fields. Use the <code>search.Type</code> 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
<code>options.columns</code>	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
<code>SSS_MISSING_REQD_ARGUMENT</code>	{1}: Missing a required argument: {2}	Required parameter is missing.

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](#).

```
...
var fieldLookUp = search.lookupFields({
    type: 'SALESORDER',
    id: '87',
    columns: ['entity', 'subsidiary', 'name', 'currency']
});
...
```

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
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

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](#).

```
...
search.lookupFields.promise({
    type : 'employee',
    id : -5,
    columns : 'email'
})
.then(function (result) {
    log.debug("Completed: " + result);
    // do something after completion
})
.catch(function onRejected(reason) {
    // do something on rejection
});
```

...

search.createColumn(options)

Method Description	Creates a new search column as a search.Column object.
Returns	search.Column
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
options.name	string	Required	Name of the search column. See Column.name .	Version 2015 Release 2
options.join	string	Optional	Join ID for the search column. See Column.join .	Version 2015 Release 2
options.summary	search.Summary	Optional	Summary type for the column. See search.Summary and Column.summary .	Version 2015 Release 2
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	search.Sort	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.

Error Code	Message	Thrown If
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

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](#).

```
.....
var currencyColumn = search.createColumn({
    name: 'currency',
    sort: 'ASC',
});
...
...
```

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

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. See search.Operator .	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

Parameter	Type	Required / Optional	Description	Since
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](#).

```
...
var mySearchFilter = search.createFilter({
    name: 'entity',
    operator: 'ISEMPTY',
});
...
```

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. 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/search Module
Since	Version 2015 Release 2

Values

• AFTER	• IS	• NOTGREATERTHANOREQUAL TO
---------	------	----------------------------

• ALLOF	• ISEMPTY	• NOTLESSTHAN
• ANY	• ISNOT	• NOTLESSTHANOREQUALTO
• ANYOF	• ISNOTEMPTY	• NOTON
• BEFORE	• LESSTHAN	• NOTONORAFTER
• BETWEEN	• LESSTHANOREQUALTO	• NOTONORBEOFRE
• CONTAINS	• NONEOF	• NOTWITHIN
• DOESNOTCONTAIN	• NOTAFTER	• ON
• DOESNOTSTARTWITH	• NOTALLOF	• ONORAFTER
• EQUALTO	• NOTBEFORE	• ONORBEFORE
• GREATERTHAN	• NOTBETWEEN	• STARTSWITH
• GREATERTHANOREQUALTO	• NOTEQUALTO	• WITHIN
• HASKEYWORDS	• NOTGREATERTHAN	

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](#).

```
...
var mySearchFilter = search.createFilter({
    name: 'entity',
    operator: 'ISEMPTY',
});
...
...
```

search.Sort

Enum Description	Enumeration that holds the values for supported sorting directions used with search.createColumn(options).
	<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

- ASC
- DESC
- NONE

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 . 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/search Module
Since	Version 2015 Release 2

Values

- GROUP
- COUNT
- SUM
- AVG
- MIN
- MAX

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](#).

```
...
var mySearchFilter = search.createFilter({
    name: 'entity',
    summary: 'GROUP',
});
...
```

search.Type

Enum Description	Enumeration that holds the string values for record types that support search.create(options) . Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Module	N/search Module

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

Values

<ul style="list-style-type: none"> • ACCOUNT • ACCOUNTING_BOOK • ADDRESS • AMORTIZATION_SCHEDULE • AMORTIZATION_TEMPLATE • ACTIVITY • ASSEMBLY_BUILD • ASSEMBLY_UNBUILD • BILLING_CLASS • BILLING_SCHEDULE • BIN • BIN_PUTAWAY_WORKSHEET • BIN_TRANSFER • BLANKET_PURCHASE_ORDER • BUILD_ASSEMBLY • CAMPAIGN • CAMPAIGN_TEMPLATE • CASE • CASH_REFUND • CASH_SALE • CHARGE • CHECK • CLASS • COMPETITOR • CONTACT • COUPON_CODE • CREDIT_MEMO • CURRENCY • CUSTOMER • CUSTOMER_CATEGORY • CUSTOMER_DEPOSIT • CUSTOMER_PAYMENT • CUSTOMER_REFUND 	<ul style="list-style-type: none"> • GIFT_CERTIFICATE_ITEM • GLOBAL_ACCOUNT_MAPPING • GROUP • INTERCOMPANY_JOURNAL_ENTRY • INVENTORY_ADJUSTMENT • INVENTORY_COST_REVALUATION • INVENTORY_COUNT • INVENTORY_DETAIL • INVENTORY_ITEM • INVENTORY_NUMBER • INVENTORY_TRANSFER • INVOICE • ISSUE • ITEM_ACCOUNT_MAPPING • ITEM_SEARCH • ITEM_DEMAND_PLAN • ITEM_FULFILLMENT • ITEM_GROUP • ITEM_RECEIPT • ITEM_REVISION • ITEM_SUPPLY_PLAN • JOURNAL_ENTRY • KIT • LANDED_COST • LEAD • LOCATION • LOT_NUMBERED_ASSEMBLY_ITEM • LOT_NUMBERED_INVENTORY_ITEM • MANUFACTURING_COST_TEMPLATE 	<ul style="list-style-type: none"> • PROJECT_TASK • PROMOTION • PROSPECT • PURCHASE_CONTRACT • PURCHASE_ORDER • REALLOCATE_ITEMS • REQUISITION • RESOURCE_ALLOCATION • RETURN_AUTHORIZATION • REVENUE_COMMITMENT • REVENUE_COMMITMENT_REVERSAL • REVENUE_RECOGNITION_SCHEDULE • REVENUE_RECOGNITION_TEMPLATE • SALES_ORDER • SALES_TAX_ITEM • SCHEDULED_SCRIPT_INSTANCE • SERIALIZED_ASSEMBLY_ITEM • SERIALIZED_INVENTORY_ITEM • SERVICE • SOLUTION • STATISTICAL_JOURNAL_ENTRY • SUBSIDIARY • SUBTOTAL • TASK • TAX_CONTROL_ACCOUNT • TAX_GROUP • TAX_PERIOD • TAX_TYPE • TERM • TIME
--	--	--

• CUSTOM_LIST	• MANUFACTURING_PLANNED_TIME	• TOPIC
• DEPARTMENT	• MANUFACTURING_OPERATION_TASK	• TRANSACTION_SEARCH
• DEPOSIT	• MANUFACTURING_ROUTING	• TRANSFER_ORDER
• DEPOSIT_APPLICATION	• MARKUP	• UNIT_OF_MEASURE
• DESCRIPTION	• MESSAGE	• VENDOR
• DISCOUNT	• MULTIBOOK_ACCOUNTING_TRANSACTION	• VENDOR_BILL
• DOWNLOAD_ITEM	• NEXUS	• VENDOR_CATEGORY
• EMAIL_TEMPLATE	• NON_INVENTORY_PART	• VENDOR_CREDIT
• EMPLOYEE	• NOTE	• VENDOR_PAYMENT
• ENTITY	• OPPORTUNITY	• VENDOR_RETURN_AUTHORIZATION
• ESTIMATE_QUOTE	• OTHER_CHARGE_ITEM	• WEB_SITE_SETUP
• EVENT	• OTHER_NAME	• WORK_ORDER
• EXPENSE_CATEGORY	• PARTNER	• WORK_ORDER_CLOSE
• EXPENSE_REPORT	• PAYCHECK_JOURNAL	• WORK_ORDER_COMPLETION
• FOLDER	• PAYMENT	• WORK_ORDER_ISSUE
• GIFT_CERTIFICATE	• PAYROLL_ITEM	
	• PHONE_CALL	
	• PRICE_LEVEL	
	• PROJECT_JOB	
	• PROJECT_EXPENSE_TYPE	

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 \(SuiteSignOn\)](#).

- [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

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 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.
Note:	To use this method, Outbound Single Sign-on and web services must be enabled in your account. To enable these features, go to Setup > Company > Enable Features. On the SuiteCloud tab, in the Manage Authentication section, select the SuiteSignOn check box. In the SuiteTalk section, select the Web Services check box. Click Save.
Returns	URL, OAuth token, and any integration variables as a string
Supported Script Types	Portlet scripts, user event scripts, and Suitelets
Governance	20 units
Module	N/sso Module
Since	Version 2015 Release 2

Parameters

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description	Since
options.suiteSignOnId	string	required	The scriptId specified on the SuiteSignOn record. To see a list of IDs for SuiteSignOn records, go to the SuiteSignOn	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			<p>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 use SuiteBundler to deploy your scripts into other accounts.</p>	

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. Note: The suiteSignOnId input parameter must be a scriptId and not a internal id.
SSO_CONFIG_REQD	The SuiteSignOn object {1} is not configured for use with this script. You must specify the script as a connection point for this SuiteSignOn.	The suiteSignOnId input parameter is missing.

Syntax

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](#).

```
...
var suiteSignOnRecordId = 1;
var url = sso.generateSuiteSignOnToken('customsso1');
...
```

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.

- [N/task Module Members](#)

- task.ScheduledScriptTask
- task.ScheduledScriptTaskStatus
- task.MapReduceScriptTask
- task.MapReduceScriptTaskStatus
- task.CsvImportTask
- task.CsvImportTaskStatus
- task.EntityDeduplicationTask
- task.EntityDeduplicationTaskStatus
- task.WorkflowTriggerTask
- task.WorkflowTriggerTaskStatus
- 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 properties and 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

Member Type	Name	Return Type / Value Type	Description
			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.WorkflowTriggerTask	Object	Encapsulates all the properties required to asynchronously initiate a workflow. Use WorkflowTriggerTask to create a task that initiates an instance of a specific workflow.
	task.WorkflowTriggerTaskStatus	Object	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.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.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

Member Type	Name	Return Type / Value Type	Description
			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 task.EntityDeduplicationTask .
	task.DedupeMode	enum	Enumeration that holds the string values for available deduplication modes when merging duplicate records with task.EntityDeduplicationTask .
	task.DedupeEntityType	enum	Enumeration that holds the string values for entity types for which you can merge duplicate records with task.EntityDeduplicationTask .
	task.MapReduceStage	enum	Enumeration that holds the string values for possible stages in task.MapReduceScriptTask 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.
	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

Member Type	Name	Return Type / Value Type	Description
			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 .
	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.

Member Type	Name	Return Type / Value Type	Description
			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.
	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

Member Type	Name	Return Type / Value Type	Description
			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.
	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.

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.

Member Type	Name	Return Type / Value Type	Description
	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

The following example creates and submits a map reduce script job and sends an email on failure.

```
/*
 * @NApiVersion 2.x
 */
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
            });
            mrTask.scriptId = mapReducescriptId;
            mrTask.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();
    }
)
```

});

task.ScheduledScriptTask

Object Description	Encapsulates all the properties of scheduled script task in SuiteScript. Use this object to place a scheduled script deployment into the NetSuite scheduling queue. To use the ScheduledScriptTask Object: <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 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

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](#).

```
...
var scriptTask = task.create({taskType: task.TaskType.SCHEDULED_SCRIPT});
scriptTask.scriptId = 1234;
scriptTask.deploymentId = 'customdeploy1';
scriptTask.params = {searchId: 'custsearch_456'};
var scriptTaskId = scriptTask.submit();
...

```

ScheduledScriptTask.submit()

Method Description	Directs NetSuite to place a scheduled script deployment into the NetSuite scheduling queue and returns a unique ID for the task. Scheduled scripts must meet the following requirements: <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.
---------------------------	--

	<ul style="list-style-type: none"> 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

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](#).

```
...
var scheduledScriptTaskId = scriptTask.submit();
...
```

ScheduledScriptTask.scriptId

Property Description	Internal ID (as a number), or script ID (as a string), for the script record associated with a <code>task.ScheduledScriptTask</code> object.
Type	number string
Module	N/task 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/task Module Script Sample](#).

```
...
var scheduledscriptId = 34;
...
```

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

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](#).

```
...
scheduledTask.deploymentId = 1;
...
```

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

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](#).

```
...
scriptTask.params = {searchId: 'custsearch_456'};
...
```

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.
Supported Script Types	Server-side scripts

Module	N/task 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/task Module Script Sample](#).

```
...
var res = task.checkStatus(scriptTaskId);
log.debug('Initial status: ' + res.status);
...
```

ScheduledScriptTaskStatus.scriptId

Property Description	Internal ID for a script record associated with a specific task.ScheduledScriptTask Object. Use this ID to get more details about the script record for the scheduled task.
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

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](#).

```
...
log.audit('Initial status: ' + status.scriptId);
...
```

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

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](#).

```
...
log.audit('Deployment ID: ' + status.scriptId);
...
```

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

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](#).

```
...
log.audit('Status: ' + summary.status);
...
```

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 task.create(options) to create the <code>MapReduceScriptTask</code> object.

	<ul style="list-style-type: none"> • 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. <p>For more information about scheduled scripts in NetSuite, see Map/Reduce Script Type.</p>
Supported Script Types	Server-side scripts
Module	N/task 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/task Module Script Sample](#).

```
...
var mrTask = task.create({taskType: task.TaskType.MAP_REDUCE});
mrTask.scriptId = mapReducescriptId;
mrTask.deploymentId = 1;
var mrTaskId = mrTask.submit();
...
```

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

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](#).

```
...
var mrTaskId = mrTask.submit();
...
```

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

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](#).

```
...
var mapReducescriptId = 34;
...
```

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

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](#).

```
...
mrTask.deploymentId = 1;
...
```

MapReduceScriptTask.params

Property Description	Object that represents key/value pairs that override static script parameter field values on the script deployment record.
-----------------------------	--

Use these parameters on a `task.MapReduceScriptTask` object to programmatically pass values to the script deployment. For more information about script parameters, see the help topic [Creating Script Parameters Overview](#).

Type	object
Module	N/task 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/task Module Script Sample](#).

```
...
mrTask.params = {doSomething: true};
...
```

task.MapReduceScriptTaskStatus

Object Description	Encapsulates the properties and status of a map/reduce script deployment placed into the NetSuite task queue. Use <code>task.checkStatus(options)</code> with the unique ID for the map/reduce script task to get the <code>MapReduceScriptTaskStatus</code> object.
Supported Script Types	Server-side scripts
Module	N/task 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/task Module Script Sample](#).

```
...
var summary = task.checkStatus(scriptTaskId);
if (summary.stage === task.MapReduceStage.SUMMARIZE)
    log.audit('Almost done...');

...
```

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

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](#).

```
...
var completion = taskStatus.getPercentageCompleted();
log.audit('Percentage Completed: ' + completion);
...
```

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

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](#).

```
...
var summary = taskStatus.getPendingMapCount();
log.audit('Pending Map Count: ' + summary);
...
```

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

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](#).

```
...
var summary = taskStatus.getTotalMapCount();
log.audit('Total Map Count: ' + summary);
...
```

MapReduceScriptTaskStatus.getPendingMapSize()

Method Description	Returns the total number of bytes not yet processed by the map stage, as a component of total size, of a task.MapReduceScriptTask . Use the MapReduceScriptTaskStatus.stage property to get the current stage.
Returns	number
Supported Script Types	Server-side scripts
Governance	25 units
Module	N/task 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/task Module Script Sample](#).

```
...
var summary = taskStatus.getPendingMapSize();
log.audit('Pending Map Size: ' + summary);
...
```

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

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](#).

```
...
var summary = taskStatus.getPendingReduceCount();
log.audit('Pending Reduce Count: ' + summary);
...
```

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

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](#).

```
...
var summary = taskStatus.getTotalReduceCount();
log.audit('Reduce Count: ' + summary);
...
```

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

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](#).

```
...
var summary = taskStatus.getPendingReduceSize();
log.audit('Pending Reduce Size: ' + summary);
...
```

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

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](#).

```
...
var summary = task.checkStatus(scriptTaskId);
var total = summary.getPendingOutputCount();
log.audit('Count', total);
...
```

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

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](#).

```
...
var summary = task.checkStatus(scriptTaskId);
var total = summary.getTotalOutputCount()
log.audit('Size', total);
...
```

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

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](#).

```
...
var summary = task.checkStatus(scriptTaskId);
var total = summary.getTotalOutputCount()
log.audit('Total Entries Passed to Output', total);
...
```

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

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](#).

```
...
var summary = task.checkStatus(scriptTaskId);
var total = summary.getCurrentTotalSize()
log.audit('Size of Remaining Data to Process', total);
...
```

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

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](#).

```
...
var summary = task.checkStatus(scriptTaskId);
log.audit('Script ID', summary.scriptId);

...
```

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

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](#).

```
...
var summary = task.checkStatus(scriptTaskId);
log.audit('Deployment ID', summary.deploymentId);
```

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

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](#).

```
...
var summary = task.checkStatus(scriptTaskId);
log.audit('Status', summary.status);
```

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

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](#).

```
...
var summary = task.checkStatus(scriptTaskId);
if (summary.stage === task.MapReduceStage.SUMMARIZE)
    log.audit('Almost done...');

...
```

task.CsvImportTask

Object Description	<p>Encapsulates the properties of a CSV import task. Use the methods and properties for this object to submit a CSV import task into the task queue and asynchronously import record data into NetSuite.</p> <p>Use the <code>CsvImportTask</code> Object to perform the following types of tasks:</p> <ul style="list-style-type: none"> 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. <p>Use the following process to import CSV data with <code>CsvImportTask</code>:</p> <ul style="list-style-type: none"> In the NetSuite UI, run the Import Assistant to set up the CSV mapping and import options. You must run the Import Assistant to set up the necessary mapping for the CSV import. You can use a sample file or files to set up the mapping. Note the following information: <ul style="list-style-type: none"> Script ID for import map. Any required linked files. For more information, see the help topic Importing CSV Files with the Import Assistant. Use <code>task.create(options)</code> to create the <code>CsvImportTask</code> object. Use the <code>CsvImportTask</code> object properties to set the script and deployment properties. Use <code>CsvImportTask.submit()</code> to submit the import task to the NetSuite task queue. Use the properties for the <code>task.CsvImportTaskStatus</code> object to get the status of the import process. <p>Use the following guidelines with the <code>CsvImportTask</code> Object:</p> <ul style="list-style-type: none"> CSV imports performed within scripts are subject to the existing application limit of 25,000 records.
---------------------------	---

	<ul style="list-style-type: none"> 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. <p>Even if you set options to use multiple threads or queues for an import job and then save the import map, these settings are not available to <code>CsvImportTask</code>. When this object submits a CSV import job based on the saved import map, a single thread and single queue are used.</p>
Supported Script Types	Server-side scripts
Module	N/task 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/task Module Script Sample](#).

```
...
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();
...

```

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

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](#).

```
...
var csvImportTaskId = csvTask.submit();
...
```

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

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](#).

```
...
var f = file.load('SuiteScripts/custjoblist.csv');
scriptTask.importFile = f;
...
```

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

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](#).

```
...
scriptTask.mappingId = 51;
```

...

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.
	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 .
Type	number
Module	N/task 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/task Module Script Sample](#).

```
...
scriptTask.queueId = 2;
...
```

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

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](#).

```
...
csvTask.name = 'Import Entities'
...
```

CsvImportTask.linkedFiles

Property Description	A map of key/value pairs that sets the data to be imported in a linked file for a multi-file import job, by referencing a file in the file cabinet or the raw CSV data to import.
-----------------------------	---

The key is the internal ID of the record sublist for which data is being imported and the value is either a [file.File](#) object or the raw CSV data to import.

You can assign multiple types of values to the `linkedFiles` property.

Type	Object
Module	N/task 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/task Module Script Sample](#).

```
...
scriptTask.linkedFiles = {'addressbook': 'street,city\nval1,val2', 'purchases': file.load('Suit
eScripts/other.csv')};
...
```

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 <code>CsvImportTaskStatus</code> object.
Supported Script Types	Server-side scripts
Module	N/task 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/task Module Script Sample](#).

```
...
var csvTaskStatus = task.checkStatus(csvTaskId);
if (csvTaskStatus.status === task.TaskStatus.FAILED)
...
```

CsvImportTaskStatus.status

Property Description	Status for a CSV import 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

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](#).

```
...
var summary = task.checkStatus(scriptTaskId);
log.audit('Status', summary.status);
...
```

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 allows 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 EntityDeduplicationTask Object:</p> <ul style="list-style-type: none"> • Use <code>task.create(options)</code> to create the EntityDeduplicationTask 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 EntityDeduplicationTask 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.
Supported Script Types	Server-side scripts
Module	N/task 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/task Module Script Sample](#).

```
...
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();
...

```

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

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](#).

```
...  
var dedupeTaskId = dedupeTask.submit();  
...
```

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	task.DedupeEntityType
Module	N/task 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/task Module Script Sample](#).

```
...  
dedupeTask.entityType = task.DedupeEntityType.CUSTOMER;  
...
```

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	number
Module	N/task 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/task Module Script Sample](#).

```
...  
dedupeTask.masterSelectionMode = task.MasterSelectionMode.SELECT_BY_ID;  
dedupeTask.masterRecordId = 107;
```

...

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

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](#).

```
...
dedupeTask.masterSelectionMode = task.MasterSelectionMode.MOST_RECENT_ACTIVITY;
...
```

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

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](#).

```
...
dedupeTask.dedupeMode = task.DedupeMode.MERGE;
...
```

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

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](#).

```
...
dedupeTask.recordIds = ['107', '110'];
...
```

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.
Supported Script Types	Server-side scripts
Module	N/task 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/task Module Script Sample](#).

```
...
var dedupeTaskStatus = task.checkStatus(taskId);
if (dedupeTaskStatus.status === task.TaskStatus.FAILED)
...
...
```

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

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](#).

```
...
var summary = task.checkStatus(scriptTaskId);
log.audit('Status', summary.status);
...
```

task.WorkflowTriggerTask

Object Description	Encapsulates all the properties required to asynchronously initiate a workflow. Use the WorkflowTriggerTask Object to create a task that initiates an instance of the specified workflow. The task is placed in the scheduling queue, and the workflow instance is initiated once the task reaches the top of the queue. To use the WorkflowTriggerTask Object: <ul style="list-style-type: none">• Use task.create(options) to create the WorkflowTriggerTask Object.• Use WorkflowTriggerTask.recordType to set the record type of the workflow base record.• Use WorkflowTriggerTask.recordId to set the internal ID of the base record for the workflow.• Use WorkflowTriggerTask.workflowId to set the internal ID of the workflow that you want to run on the record specified by the <code>recordId</code>.• Optionally, use WorkflowTriggerTask.params to specify default values for workflow fields.• Use WorkflowTriggerTask.submit() to submit the asynchronous workflow initiation task to the NetSuite task queue.• Use the properties for the WorkflowTriggerTaskStatus.status object to get the status of the workflow execution. Use the following guidelines with the WorkflowTriggerTask Object: <ul style="list-style-type: none">• WorkflowTriggerTask.submit() 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.
Supported Script Types	Server-side scripts
Module	N/task 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/task Module Script Sample](#).

```
...
var workflowTask = task.create({taskType: task.TaskType.WORKFLOW_TRIGGER});
workflowTask.recordType = 'customer';
workflowTask.recordId = 107;
workflowTask.workflowId = 3;
var taskId = workflowTask.submit();
...
```

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

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](#).

```
...
var workflowTriggerTask = workflowTask.submit();
...
```

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

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](#).

```
...  
workflowTask.recordType = 'customer';  
...
```

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

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](#).

```
...  
workflowTask.recordId = 107;  
...
```

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

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](#).

```
...
workflowTask.workflowId = 3;
...
```

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

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](#).

```
...
task.params = {context: portlet}
...
```

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.
Supported Script Types	Server-side scripts
Module	N/task 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/task Module Script Sample](#).

```
...
var workflowTaskStatus = task.checkStatus(taskId);
if (workflowTaskStatus.status === task.TaskStatus.FAILED)
...
...
```

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

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](#).

```
...
var summary = task.checkStatus(scriptTaskId);
log.audit('Status', summary.status);

...
```

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, and execute asynchronous workflows.
Returns	<code>task.ScheduledScriptTask</code> <code>task.MapReduceScriptTask</code> <code>task.CsvImportTask</code> <code>task.EntityDeduplicationTask</code> <code>task.WorkflowTriggerTask</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.taskType</code>	<code>task.TaskType</code>	Required	The type of task object to create. Use a <code>task.TaskType</code> enum value.	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/task Module Script Sample](#).

```
...
var mrTask = task.create({taskType: task.TaskType.MAP_REDUCE});
...
```

task.checkStatus(options)

Method Description	Returns a task status object associated with a specific task ID.
Returns	task.ScheduledScriptTaskStatus task.MapReduceScriptTaskStatus task.CsvImportTaskStatus task.EntityDeduplicationTaskStatus task.WorkflowTriggerTaskStatus
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.taskId	task.ScheduledScriptTask task.MapReduceScriptTask task.CsvImportTask task.EntityDeduplicationTask task.WorkflowTriggerTask	Required	Unique ID for the task that was generated by task.create(options) .	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/task Module Script Sample](#).

```
...
var taskStatus = task.checkStatus(mrTaskId);
...
```

task.TaskType

Enum Description	Enumeration that holds the string values for the types of task objects supported by the N/task Module , that you can create with task.create(options) .
-------------------------	---

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

Module	N/task Module
Since	Version 2015 Release 2

Values

- SCHEDULED_SCRIPT
- MAP_REDUCE
- CSV_IMPORT
- ENTITY_DEDUPLICATION
- WORKFLOW_TRIGGER

Syntax

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](#).

```
...
var mrTask = task.create({
    taskType: task.TaskType.MAP_REDUCE
});
...
...
```

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.</p> <p>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>EntityDuplicationTaskStatus.status</code> • <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

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](#).

```
...
if (status == task.TaskStatus.COMPLETE || status == task.TaskStatus.FAILED)
...
```

task.MasterSelectionMode

Enum Description	Enumeration that holds the string values for supported master selection modes when merging duplicate records with task.EntityDeduplicationTask . Use this enum for the EntityDeduplicationTask.masterSelectionMode property. For more information about these values, see the help topic Merging or Deleting Duplicate Records . Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Module	N/task Module
Since	Version 2015 Release 2

Values

- CREATED_EARLIEST
- MOST_RECENT_ACTIVITY
- MOST_POPULATED_FIELDS
- SELECT_BY_ID

Syntax

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](#).

```
...
dedupeTask.masterSelectionMode = task.MasterSelectionMode.MOST_RECENT_ACTIVITY;
...
```

task.DedupeMode

Enum Description	Enumeration that holds the string values for the available deduplication modes when merging duplicate records with task.EntityDeduplicationTask . Use this enum for the EntityDeduplicationTask.dedupeMode property. 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

- MERGE
- DELETE
- MAKE_MASTER_PARENT
- MARK_AS_NOT_DUPES

Syntax

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](#).

```
...
dedupeTask.dedupeMode = task.DedupeMode.MERGE;
...
```

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 property. Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Module	N/task Module
Since	Version 2015 Release 2

Values

- CUSTOMER

- CONTACT
- VENDOR
- PARTNER
- LEAD
- PROSPECT

Syntax

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](#).

```
...
dedupeTask.entityType = task.DedupeEntityType.CUSTOMER;
...
```

task.MapReduceStage

Enum Description	Enumeration that holds the string values for possible stages in task.MapReduceScriptTask for a map/reduce script. This enum is returned by MapReduceScriptTaskStatus.stage . 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

- GET_INPUT
- MAP
- SHUFFLE
- REDUCE
- SUMMARIZE

Syntax

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](#).

```
...
if (summary.stage === tak.MapReduceStage.SUMMARIZE)
...
```

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. The void type is based on the account preferences and transaction type. For more information, see the help topic [Voiding, Deleting, or Closing Transactions](#).

- 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
 */
require(['N/transaction', 'N/config', 'N/record'],
    function(transaction, config, record) {
        function voidSalesOrder() {
            var accountingConfig = config.load({
                type: config.Type.ACCOUNTING_PREFERENCES
            });
            accountingConfig.setValue({
                name: '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. Important: After you void a transaction, you cannot make changes to the transaction that impact the general ledger.
Returns	An ID returned as a number. <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 type argument passed is not valid or the record type is not voidable.
THAT_RECORD_DOES_NOT_EXIST		The id argument passed is not valid.
SSS_MISSING_REQD_ARGUMENT		The type or id argument is missing.

Syntax

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](#).

```
...
var voidSalesOrderId = transaction void({
    type: transaction.Type.SALES_ORDER,
    id: salesOrderId
});
```

transaction.void.promise(options)

Method Description	Method used to void a transaction record object asynchronously and return an id that indicates the type of void performed. The type of void performed depends on the targeted account's preference settings. Important: After you void a transaction, you cannot make changes to the transaction that impact the general ledger. Note: For information about the parameters and errors thrown for this method, see transaction.void(options) . For additional information on promises, see Promise object .
Returns	An id returned as a number.

	<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	<code>transaction.void(options)</code>
Supported Script Types	All client-side scripts
Governance	10 units
Module	N/transaction 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 promise script example, see [Promise object](#).

```
...
var voidSalesOrderId = transaction.void.promise({
    type: record.Type.SALES_ORDER,
    id: salesOrderId
});
```

transaction.Type

Enum Description	Enumeration that holds the string values for supported transaction 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.
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

Transaction Record	Supported Void Type
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

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](#).

```
...
var voidSalesOrderId = transaction.void({
    type: transaction.Type.SALES_ORDER,
    id: salesOrderId
});
```

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/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 Sample

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  
        };  
  
        dialog.select([button1, button2]).then(function(result) {  
            console.log("Selected value: " + result);  
        }).catch(function(error) {  
            console.log("Error: " + error);  
        });  
    });
```

```

        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

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 Sample](#).

```

...
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);
...

```

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

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

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 Sample](#).

```

...
function success(result) { console.log('Success with value: ' + result) }
function failure(reason) { console.log('Failure: ' + reason) }

dialog.alert({
    title: 'Confirmation',
    message: 'Click OK or Cancel to continue.'
}).then(success).catch(failure);

```

...

dialog.create(options)

Method Description	Creates a dialog with specified buttons.
Returns	Promise object . To run a callback function when a button is pressed, pass a function to the then portion of the Promise object. The value of the button pressed is passed to the callback.
Supported Script Types	Client scripts
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.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

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 Sample](#).

```
...
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);
...

```

N/ui/message module

Load the message module to display a message at the top of the screen under the menu bar.

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.

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.
Supported Script Types	Client scripts
Module	N/ui/message module
Since	Version 2016 Release 1

Message Object Members

Member Type	Name	Return Type / Value Type	Description
Method	Message.hide()	void	Hides the message.
	Message.show()	void	Shows the message.

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

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](#).

```

...
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
...
```

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

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](#).

```
...
var myMsg = message.create({
    title: "My Title 2",
    message: "My Message 2",
    type: message.Type.INFORMATION
});
myMsg.show({ duration : 1500 });
...
```

message.create(options)

Method Description	Creates a message that can be displayed or hidden near the top of the page.
Returns	message.Message.
Supported Script Types	Client scripts
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.type	message.Type	required	The message type.	Version 2016 Release 1
options.title	string	optional	The message title. This value defaults to an empty string.	Version 2016 Release 1
options.message	string	optional	The content of the message. This value defaults to an empty string.	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/ui/message Module Script Sample](#).

```
...
var myMsg = message.create({
    title: "My Title",
    message: "My Message",
    type: message.Type.CONFIRMATION
});
...
```

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.

Value	Color
ERROR	A red background with an X icon.

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/message Module Script Sample](#).

```
...
var myMsg = message.create({
    title: "My Title",
    message: "My Message",
    type: message.Type.CONFIRMATION
});
myMsg.show();
...
```

N/ui/serverWidget Module

Load the serverWidget module when you want to work with the user interface within NetSuite. You can use certain server scripts, such as 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 the 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.
	serverWidget.Tab	Object	Encapsulates NetSuite tabs and subtabs.
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.
	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.

Member Type	Name	Return Type / Value Type	Description
	serverWidget.FormPageLinkType	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 <code>Form.addPageLink(options)</code> .
	serverWidget.LayoutJustification	string (read-only)	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.
	serverWidget.ListStyle	string (read-only)	Holds the string values for supported list styles. This enum is used to set the value of the <code>List.style</code> property.
	serverWidget.LayoutJustification	string (read-only)	Holds the string values for supported sublist display types. This enum is used to set the value of the <code>Sublist.displayType</code> property.
	serverWidget.SublistType	string (read-only)	Holds the string values for valid sublist types. This enum is used to define the type parameter when <code>Form.addSublist(options)</code> 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.
	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.

Member Type	Name	Property Type / Method Return Type	Description
	Assistant.getFieldIdsByFieldGroup(fieldGroup)	string[]	Gets all field IDs in the given assistant field group.
	Assistant.getLastAction()	string	Gets the last action submitted by the user.
	Assistant.getLastStep()	serverWidget.AssistantStep	Gets the last step taken by the user.
	Assistant.getNextStep()	serverWidget.AssistantStep	Gets the next step prompted by the assistant.
	Assistant.getStep(options)	serverWidget.AssistantStep	Get the a step given its ID.
	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 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.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

Member Type	Name	Property Type / Method Return Type	Description
			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	Message.hide()	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.
	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.
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 for the field.
	Field.padding	number	The number of empty vertical spaces above the field
	Field.richTextHeight	number	The height of a rich text field.
	Field.richTextWidth	number	The width of a rich text field.
	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.

Member Type	Name	Property Type / Method Return Type	Description
	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.
	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.updateDefaultValues(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.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.style	string	Sets the display style for this list.
	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	<code>ListColumn.addParamToURL(options)</code>	<code>serverWidget.ListColumn</code>	Adds a URL parameter (optionally defined per row) to the list column's URL.
	<code>ListColumn.setURL(options)</code>	<code>serverWidget.ListColumn</code>	Sets the base URL for the list column.
Property	<code>ListColumn.label</code>	<code>string</code>	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	<code>Sublist.addButton(options)</code>	<code>serverWidget.Button</code>	Adds a button to a sublist.
	<code>Sublist.addField(options)</code>	<code>serverWidget.Field</code>	Add a field to a sublist.
	<code>Sublist.addMarkAllButtons()</code>	<code>serverWidget.Button</code>	Adds a Mark All or Unmark All button.
	<code>Sublist.addRefreshButton()</code>	<code>serverWidget.Button</code>	Adds a Reset button.
	<code>Sublist.getSublistValue(options)</code>	<code>serverWidget.Button</code>	Adds a button to a sublist.
	<code>Sublist.setSublistValue(options)</code>	<code>void</code>	Set the value of a field on the list
	<code>Sublist.updateTotallingFieldId(options)</code>	<code>serverWidget.Sublist</code>	Updates the ID of a field designated as a totalling column, which is used to calculate and display a running total for the sublist.
	<code>Sublist.updateUniqueFieldId(options)</code>	<code>serverWidget.Sublist</code>	Updates a field ID that is to have unique values across the rows in the sublist.
Property	<code>Sublist.displayType</code>	<code>string</code>	The display style for a sublist.
	<code>Sublist.helpText</code>	<code>string</code>	The inline help text for a sublist.
	<code>Sublist.label</code>	<code>string</code>	The label for a sublist.
	<code>Sublist.lineCount</code>	<code>number</code>	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/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 Assistant 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.
Supported Script Types	Server-side scripts
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
```

Assistant.addField(options)

Method Description	Adds a field to an assistant.
Returns	serverWidget.Field object
Supported Script Types	Server-side scripts
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. Note: If you have set the type parameter to SELECT, and you want to add custom options to the select field, you must set source to NULL. Then, when a value is specified, the value will populate the options from the source.	Version 2015 Release 2
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
			Note: If you want to add custom options on a select field, you must set the source parameter to NULL.	
options.container	string	optional	The internal ID of the field group to place this field in.	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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addField({
    id : 'idname',
    type : serverWidget.FieldType.TEXT,
    label : 'Sample label'
});
```

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 FieldGroup.isCollapsed and FieldGroup.isCollapsible properties.
Returns	serverWidget.FieldGroup object
Supported Script Types	Server-side scripts
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

The following code 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.addFieldGroup({
    id : 'idname',
    label : 'Sample label'
});
```

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	Server-side scripts
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

The following code 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.addStep({
    id : 'idname',
    label : 'Sample label'
});
...
```

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	Server-side scripts
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

The following code 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.addSublist({
    id : 'idname',
    label : 'Sample label',
    type : serverWidget.SublistType.LIST
});
```

...

Assistant.getField(options)

Method Description	Returns a field object on an assistant page.
Returns	serverWidget.Field
Supported Script Types	Server-side scripts
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

The following code 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.addField({
    id : 'idname',
    type : serverWidget.FieldType.TEXT,
    label : 'Sample label'
});
var field = assistant.getField({
    id: 'idname'
});
...
```

Assistant.getFieldGroup(options)

Method Description	Returns a field group object on an assistant page.
Returns	serverWidget.FieldGroup object
Supported Script Types	Server-side scripts
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

The following code 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.addFieldGroup({
    id : 'idname',
    label : 'Sample label'
});
var fieldgroup = assistant.getFieldGroup({
    id: 'idname'
});
...
...
```

Assistant.getFieldGroupIds()

Method Description	Retrieves all the internal IDs for field groups in an assistant.
Returns	string[]
Supported Script Types	Server-side scripts
Governance	None
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addFieldGroup({
    id : 'idname',
    label : 'Sample label'
});
var fieldgroupid = assistant.getFieldGroupIds();
...
...
```

Assistant.getFieldIds()

Method Description	Gets all the internal IDs for fields in an assistant.
Returns	string[]
Supported Script Types	Server-side scripts
Governance	None
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addField({
    id : 'idname',
    label : 'Sample label'
});
var fieldid = assistant.getFieldIds();
...
```

Assistant.getFieldIdsByFieldGroup(fieldGroup)

Method Description	Gets all field IDs in the given assistant field group.
Returns	string[]
Supported Script Types	Server-side scripts
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

The following code 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.addFieldGroup({
        id : 'fieldgroupid',
        label : 'Sample label'
    });
    assistant.addField({
        id : 'idname',
        label : 'Sample label'
    });
    var fieldid = assistant.getFieldIdsByFieldGroup({
        fieldGroup : 'fieldgroupid'
    });
    ...
}
```

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	Server-side scripts
Governance	None
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
if (assistant.getLastAction() == serverWidget.AssistantSubmitAction.CANCEL) {
    ...
}
...
```

Assistant.getLastStep()

Method Description	Gets the step associated with the last action submitted by the user.
Returns	A serverWidget.AssistantStep object
Supported Script Types	Server-side scripts
Governance	None
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
var lastStep = assistant.getLastStep();
...
```

Assistant.getNextStep()

Method Description	Gets the next step corresponding to the user's last submitted action in the assistant. If you need information about the last step, use Assistant.getLastStep() before you use this method.
Returns	serverWidget.AssistantStep object
Supported Script Types	Server-side scripts
Governance	None
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
var nextStep = assistant.getNextStep();
...
```

Assistant.getStep(options)

Method Description	Returns a step in an assistant.
Returns	serverWidget.AssistantStep object
Supported Script Types	Server-side scripts
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

The following code 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.addStep({
    id : 'idname',
    label : 'Sample label'
});
var firststep = assistant.getStep({
    id: 'idname'
});
...
...
```

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	Server-side scripts
Governance	None
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
var numSteps = assistant.getStepCount();
...
...
```

Assistant.getSteps()

Method Description	Gets all the steps in an assistant.
Returns	serverWidget.AssistantStep[] object
Supported Script Types	Server-side scripts
Governance	None
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
var steps = assistant.getSteps();
...
```

Assistant.getSublist(options)

Method Description	Returns a sublist in an assistant.
Returns	serverWidget.Sublist object
Supported Script Types	Server-side scripts
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

The following code 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.addSublist({
    id : 'idname',
    label : 'Sample label',
    type : serverWidget.SublistType.LIST
});
var sublist = assistant.getSublist({
    id : 'idname'
});
...
...
```

Assistant.getSublistIds()

Method Description	Gets the IDs for all the sublists in an assistant.
Returns	string[]
Supported Script Types	Server-side scripts
Governance	None
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addSublist({
    id : 'idname',
    label : 'Sample label',
    type : serverWidget.SublistType.LIST
});
var sublistid = assistant.getSublistIds();
...
```

Assistant.hasErrorHtml()

Method Description	Determine whether an assistant has an error message to display for the current step.
Returns	boolean true if <code>Assistant.errorHtml</code> contains a value.
Supported Script Types	Server-side scripts
Governance	None
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
if (assistant.hasErrorHtml()) {
    ...
}
...
```

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	Server-side scripts
Governance	None
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
if (assistant.isFinished()) {
    ...
}
...
```

Assistant.sendRedirect(options)

Method Description	Manages redirects in an assistant.
---------------------------	------------------------------------

This method also addresses the case in which one assistant redirects to another assistant. In this scenario, the second assistant must return to the first assistant if the user Cancels or Finishes. This method, when used in the second assistant, ensures that users are redirected back to the first assistant.

Returns	void
Supported Script Types	Server-side scripts
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

The following code 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: 'Small Business Setup Assistant',
    hideNavBar: true
});
if (request.method === 'POST') {
    if (assistant.getLastAction() === 'finish') {
        assistant.finishedHtml = 'Completed!';
        assistant.sendRedirect({
            response: response
        });
    }
}
...

```

Assistant.setSplash(options)

Method Description	Defines a splash message.
Returns	void
Supported Script Types	Server-side scripts
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

The following code 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."
});
...
```

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	Server-side scripts
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

The following code 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.addField({
    id : 'idname',
    type : serverWidget.FieldType.TEXT,
    label : 'Sample label'
});
assistant.updateDefaultValues({
    idname : "New Default Value"
});
...

```

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

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](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.clientscriptId = 32;
...
```

Assistant.currentStep

Property Description	Identifies the current step. You can set any step to be the current step.
Type	serverWidget.AssistantStep (read-only)
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
```

```
});  
assistant.addStep({  
    id : 'idname',  
    label : 'Sample label'  
});  
var step = assistant.currentStep;  
...
```

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

The following code 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.errorHtml = "You have <b>not</b> filled out the required fields. Please go back.";  
...
```

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

The following code 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.finishedHtml = "Congratulations! You have successfully set up your account.";  
...
```

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	<code>boolean true false</code>
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.hideAddToShortcutsLink = true;
...
```

Assistant.hideStepNumber

Property Description	Indicates whether assistant steps are displayed with numbers. By default, the value is <code>false</code> – steps are not numbered. If set to <code>true</code> , a step number appears with each step. To change step ordering, set Assistant.isNotOrdered .
Type	<code>boolean true false</code>
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
assistant.hideStepNumber = true;
```

...

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

The following code 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.addStep({
    id : 'idname',
    label : 'Sample label'
});
assistant.addStep({
    id : 'idname2',
    label : 'Sample label 2'
});
assistant.isNotOrdered = false;
...
```

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 <code>serverWidget.createAssistant(options)</code> .
Type	<code>string</code>
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
var title = assistant.title;
...
```

serverWidget.AssistantStep

Object Description	Encapsulates a step within a custom NetSuite assistant. Create a step by calling Assistant.addStep(options) .
Supported Script Types	Server-side scripts
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
var assistantStep = assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
...
```

AssistantStep.getFieldIds()

Method Description	Gets the IDs for all the fields in a step.
Returns	string[]
Supported Script Types	Server-side scripts
Governance	None
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
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();
...

```

AssistantStep.getSublistFieldIds(options)

Method Description	Gets the IDs for all the sublist fields (line items) in a step.
Returns	string[]
Supported Script Types	Server-side scripts
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

The following code 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'
});
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'
});

```

...

AssistantStep.getLineCount(options)

Method Description	Gets the number of lines on a sublist in a step.
	Note: The first line number on a sublist is 0 (not 1).
Returns	The count of line items on a sublist as a number
	Note: if the sublist does not exist, -1 is returned.
Supported Script Types	Server-side scripts
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

The following code 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'
});
var assistantStep = assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
var sublist = assistant.addSublist({
    id : 'sublistid',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Editor'
});
var numLines = assistantStep.getLineCount({
    group : 'sublistid'
});
...

```

AssistantStep.getSubmittedSublistIds()

Method Description	Gets the IDs for all the sublists submitted in a step.
Returns	string[]

Supported Script Types	Server-side scripts
Governance	None
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
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();
...

```

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	Server-side scripts
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.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			Note: The first line number on a sublist is 0 (not 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/ui/serverWidget Module Script Sample](#).

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

AssistantStep.getValue(options)

Method Description	Gets the current value(s) of a field or mult-select field.
Returns	string[]
Supported Script Types	Server-side scripts
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

The following code 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'
});
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'
});
...
...
```

AssistantStep.helpText

Property Description	The help text for a step.
Type	string
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
var assistantStep = assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
assistantStep.helpText = Help Text Goes Here.';
...
...
```

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

The following code 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'
});
var assistantStep = assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
var id = assistantStep.id';
...
...
```

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

The following code 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'
});
var assistantStep = assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
var label = assistantStep.label;
...
...
```

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

The following code 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'
});
var assistantStep = assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
var stepNum = assistantStep.stepNumber;
...
```

serverWidget.Button

Object Description	Encapsulates a custom button. 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 .
Supported Script Types	Server-side scripts
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var button = form.addButton({
    id : 'buttonid',
    label : 'Test'
});
```

...

Button.isDisabled

Property Description	Indicates whether a button is grayed-out and disabled. The default value is <code>false</code> . If set to <code>true</code> , the button appears grayed-out in the UI and cannot be clicked. Note: This method is not supported for standard NetSuite buttons. This method can be used with custom buttons only.
Type	<code>boolean true false</code>
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var button = form.addButton({
    id : 'buttonid',
    label : 'Test'
});
button.isDisabled = true;
...
```

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 <code>true</code> , 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	<code>boolean true false</code>
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var button = form.addButton({
    id : 'buttonid',
    label : 'Test'
});
button.isHidden = true;
...

```

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.
Note:	This property is supported on custom buttons and most standard buttons.
Type	string
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var button = form.addButton({
    id : 'buttonid',
    label : 'Test'
});
var label = button.label;
...

```

serverWidget.Field

Object Description	Encapsulates a NetSuite field. To add a Field object, use Assistant.addField(options) , Form.addField(options) , or Sublist.addField(options) .
Supported Script Types	Server-side scripts
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
...
...
```

Field.addSelectOption(options)

Method Description	Adds the select options that appears in the dropdown of a field.
Returns	void
Supported Script Types	Server-side scripts
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.value	string	required	The internal ID of this select option.	Version 2015 Release 2
options.text	string	required	The label for this select option.	Version 2015 Release 2
options.isSelected	boolean true false	optional	If set to true, this option is selected by default in the UI. The default value for this parameter is false.	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/ui/serverWidget Module Script Sample](#).

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

```

Field.getSelectOptions(options)

Method Description	Obtains a list of available options on a select field. The internal ID and label of the options for a select field as name/value pairs is returned. The first 1,000 available options are returned. If you attempt to get select options on a field that is not a select field, or if you reference a field that does not exist on the form, null is returned. Note: A call to this method may return different results for the same field for different roles.
Returns	Object[]
Supported Script Types	Server-side scripts
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.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 contains is startswith.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			If not specified, defaults to the <code>contains</code> 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/ui/serverWidget Module Script Sample](#).

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

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 <code>serverWidget.Field</code> object
Supported Script Types	Server-side scripts
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

Parameter	Type	Required / Optional	Description	Since
options.showInlineForAssistant	boolean true false	optional	If set to true, the field help will display inline below the field on the assistant, and in a field help popup. The default value is false — the field help appears in a popup when the field label is clicked and does not appear inline. Note: The inline parameter is available only to <code>serverWidget.Field</code> objects that have been added to <code>serverWidget.createAssistant(options)</code> objects.	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/ui/serverWidget Module Script Sample](#).

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

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	Server-side scripts
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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateBreakType({
    breakType : serverWidget.FieldBreakType.STARTCOL
});
...

```

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	Server-side scripts
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

Parameter	Type	Required / Optional	Description	Since
options.width	number	required	The new width of the field.	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/ui/serverWidget Module Script Sample](#).

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

Field.updateDisplayType(options)

Method Description	Updates the display type for the field.
Returns	serverWidget.Field object
Supported Script Types	Server-side scripts
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	number	required	The new display type of the field. For more information about possible values, see serverWidget.FieldDisplayType .	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/ui/serverWidget Module Script Sample](#).

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

```

Field.updateLayoutType(options)

Method Description	Updates the layout type for the field.
Returns	serverWidget.Field object
Supported Script Types	Server-side scripts
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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateLayoutType({
    layoutType = serverWidget.FieldLayoutType.NORMAL
}
```

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

Field.alias

Property Description	An alternate name that you can assign to a serverWidget.Field object. By default, the alias is equal to the field's internal ID. This property is only supported on scripted fields created using the N/ui/serverWidget Module .
Type	string
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...  
var form = serverWidget.createForm({  
    title : 'Simple Form'  
});  
var field = form.addField({  
    id : 'textfield',  
    type : serverWidget.FieldType.TEXT,  
    label : 'Text'  
});  
field.alias = 'fieldId';  
...
```

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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.defaultValue = 'Insert Text Here.';
...
```

Field.id

Property Description	The field internal ID.
Type	string (read-only)
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
var fieldId = field.id;
...
```

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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.isMandatory = true;
...
```

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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
var label = field.label;
...
```

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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.URL,
    label : 'URL'
});
field.linkText = 'NetSuite';
...
```

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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.maxLength = 64;
...
```

Field.padding

Property Description	The number of empty vertical 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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.padding = 50;
...
```

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. Note: Rich Text Editing must be enabled.
Type	number
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.RICHTEXT,
    label : 'Rich Text'
});
field.richTextHeight = 50;
...
```

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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.RICHTEXT,
    label : 'Rich Text'
});
field.richTextWidth = 100;
...
```

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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
var fieldtype = field.type;
...
```

serverWidget.FieldGroup

Object Description	Encapsulates a field group on serverWidget.createAssistant(options) objects and on serverWidget.Form objects.
Supported Script Types	Server-side scripts
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
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

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	boolean <code>true</code> <code>false</code>
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
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;
...

```

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	<code>boolean true false</code>
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```

...
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;
...

```

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 true, the field group is collapsed. Only supported for fields on <code>serverWidget.createAssistant(options)</code> objects
Type	<code>boolean true false</code>
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
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;
...
```

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

The following code 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 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;
...
```

FieldGroup.label

Property Description	The label for the field group.
Type	string
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
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;
...
```

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.
Supported Script Types	Server-side scripts
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
...
```

Form.addButton(options)

Method Description	Adds a button to a form.
Returns	serverWidget.Button object
Supported Script Types	Server-side scripts
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 custpage . For example, if you add a button that appears as Update Order , the button internal ID should be something similar to custpage_updateorder .	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 be triggered on a click event.	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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});

form.addButton({
    id : 'buttonid',
    label : 'Test'
});
...

```

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. 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	Server-side scripts
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 custpage if you	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			are adding the field to an existing page.	
options.label	string	required	The label for the credential field.	Version 2015 Release 2
options.restrictToCurrentUser	boolean true false	optional	<p>Controls whether use of this credential is restricted to the same user that originally entered the credential.</p> <p>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.</p> <p>If set to <code>true</code>, the credentials apply to a single user.</p>	Version 2015 Release 2
options.restrictToDomains	string[]	optional	<p>This value can also be an array of strings representing a list of domains to which the credentials can be sent.</p> <p>The domain the credentials can be sent to. For example, '<code>www.mysite.com</code>'.</p>	Version 2015 Release 2
options.restrictToScriptIds	string	optional	The scriptId of the script that is allowed to use this credential field. For example, ' <code>customscript_my_script</code> '.	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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addCredentialField({
    id : 'username',
    label : 'Username',
    restrictToDomains : 'www.mysite.com',
```

```

    restrictToScriptIds : 'customscript_my_script',
    restrictToCurrentUser : false,
});
...

```

Form.addField(options)

Method Description	Adds a field to a form.
Returns	serverWidget.Field object
Supported Script Types	Server-side scripts
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 <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</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
options.source	string	optional	The internalId or scriptId of the source list for this field if it is a select (List/Record) or multi-select field. Note: For radio fields only, the source parameter must contain the internal ID for the field. For more information about working with radio buttons, see the help topic Working with Radio Buttons .	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
...
...
```

Form.addFieldGroup(options)

Method Description	Adds a group of fields to a form.
Returns	serverWidget.FieldGroup object
Supported Script Types	Server-side scripts
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	An internal ID for the field group.	Version 2015 Release 2
options.label	string	required	The label for this field group.	Version 2015 Release 2
options.tab	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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var fieldgroup = form.addFieldGroup({
    id : 'fieldgroupid',
    label : 'Field Group'
});
...
...
```

Form.addPageLink(options)

Method Description	Adds a link to a form.
Returns	void
Supported Script Types	Server-side scripts
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 serverWidget.FormPageLinkType enum to set the value.	Version 2015 Release 2
options.url	string	required	The URL for the link.	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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
```

```
});
form.addPageLink({
    type : serverWidget.FormPageLinkType.CROSSLINK,
    title : 'NetSuite',
    url : 'http://www.netsuite.com'
})
...
}
```

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	Server-side scripts
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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addButton({
    label : 'Reset Button'
});
...
```

Form.addSecretKeyField(options)

Method Description	Add a secret key field to the form.
Returns	serverWidget.Field object
Supported Script Types	Server-side scripts
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 credential 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 scriptId 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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addSecretKeyField({
    id : 'password',
    restrictToScriptIds : 'customscript_my_script',
    restrictToCurrentUser : false,
});
...
```

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	Server-side scripts
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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var sublist = form.addSublist({
    id : 'sublistid',
    ...
});
```

```

        type : serverWidget.SublistType.INLINEEDITOR,
        label : 'Inline Editor Sublist'
    });
...

```

Form.addButton(options)

Method Description	Adds a submit button to a form.
	Note: If the row count exceeds 25, sorting is not supported on static sublists created using this method.
Returns	serverWidget.Button object
Supported Script Types	Server-side scripts
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.label	string	optional	The label for this button. If no label is provided, the label defaults to "Save".	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/ui/serverWidget Module Script Sample](#).

```

...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addButton({
    label : 'Submit Button'
});
...

```

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	Server-side scripts
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_d</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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addSubtab({
    id : 'subtabId',
    label : 'Subtab'
});
...
```

Form.addTab(options)

Method Description	Adds a tab to a form.
Returns	serverWidget.Tab object

Supported Script Types	Server-side scripts
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 custpage. For example, if you add a subtab that appears as Purchase Details , the subtab internal ID should be something similar to custpage_purchasedetails or custpage_purchase_d	Version 2015 Release 2
options.label	string	required	The label for this tab.	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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var tab = form.addTab({
    id : 'tabId',
    label : 'Tab'
});
...
```

Form.getButton(options)

Method Description	Returns a Button object by internal ID.
Returns	serverWidget.Button object
Supported Script Types	Server-side scripts
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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var button = form.addButton({
    id : 'buttonid',
    label : 'Test'
});
var button = form.getButton({
    id : 'buttonid'
});
...

```

Form.getField(options)

Method Description	Returns a Field object by internal ID.
Returns	serverWidget.Field object
Supported Script Types	Server-side scripts
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.	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			Internal IDs must be in lowercase and contain no spaces.	

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](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
var field = form.getField({
    id : 'textfield'
});
...

```

Form.getSublist(options)

Method Description	Returns a Sublist object by internal ID.
Returns	serverWidget.Sublist object
Supported Script Types	Server-side scripts
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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addSublist({
    id : 'sublistid',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
var sublist = form.getSublist({
    id : 'sublistid'
});
...

```

Form.getSubtab(options)

Method Description	Returns a subtab by internal ID.
Returns	serverWidget.Tab object
Supported Script Types	Server-side scripts
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. Internal IDs must be in lowercase and contain no spaces.	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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addSubtab({
    id : 'subtabId',
    label : 'Subtab'
});
var subtab = form.getSubtab({
    id : 'subtabId'
});
```

...

Form.getTab(options)

Method Description	Returns a tab object from its internal ID.
Returns	serverWidget.Tab object.
Supported Script Types	Server-side scripts
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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addTab({
    id : 'tabId',
    label : 'Tab'
});
var tab = form.getTab({
    id : 'tabId'
});
...

```

Form.getTabs()

Method Description	Returns an array that contains all the tabs in a form.
Returns	serverWidget.Tab[] objects
Supported Script Types	Server-side scripts
Governance	None
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Parameters

Note: The options parameter is a JavaScript object.

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](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addTab({
    id : 'tabId',
    label : 'Tab'
});
var tab = form.getTabs();
...
```

Form.insertField(options)

Method Description	Inserts a field in front of another field.
Returns	void
Supported Script Types	Server-side scripts
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

The following code 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 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'
});
...

```

Form.insertSublist(options)

Method Description	Inserts a sublist in front of another sublist.
Returns	void
Supported Script Types	Server-side scripts
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
options.nextsublist	string	required	The internal ID name of the sublist you are inserting a sublist in front of.	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/ui/serverWidget Module Script Sample](#).

```

...
```

```

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'
});
...

```

Form.insertSubtab(options)

Method Description	Inserts a subtab in front of another subtab.
Returns	void
Supported Script Types	Server-side scripts
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

The following code 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 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'  

});  

...

```

Form.insertTab(options)

Method Description	Inserts a tab in front of another tab.
Returns	void
Supported Script Types	Server-side scripts
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

The following code 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 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'
});
...

```

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	Server-side scripts
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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addButton({
    id : 'buttonid',
    label : 'Test'
});
form.removeButton({
```

```
    id : 'buttonid'
});
...

```

Form.updateDefaultValues(options)

Method Description	Updates the default values of multiple fields on the form.
Returns	void
Supported Script Types	Server-side scripts
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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
form.setDefaultValue({
    textfield : 'Text Goes Here'
})
...

```

Form.clientScriptFileId

Property Description	The file cabinet ID of client script file to be used in this form.
Type	number
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.clientScriptFileId = 32;
...
```

Form.title

Property Description	The title used for the form.
Type	string
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var title = form.title;
...
```

serverWidget.List

Object Description	Encapsulates a list.
Supported Script Types	Server-side scripts
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var list = serverWidget.createList({
    title : 'Simple List'
```

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

List.addButton(options)

Method Description	Adds a button to a list.
Returns	serverWidget.Button object
Supported Script Types	Server-side scripts
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 onclick script used for this button.	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/ui/serverWidget Module Script Sample](#).

```
...  
var list = serverWidget.createList({  
    title : 'Simple List'  
});  
list.addButton({  
    id : 'buttonId',  
    label : 'Test'  
});
```

...

List.addColumn(options)

Method Description	Adds a column to a list.
Returns	serverWidget.ListColumn object
Supported Script Types	Server-side scripts
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.align	string	optional	The default value is left. The layout justification for this column. Possible values include center, right, left	Version 2015 Release 2
options.id	string	required	The internal ID of this column. The internal ID must be in lowercase, contain no spaces.	Version 2015 Release 2
options.label	string	required	The label for this column.	Version 2015 Release 2
options.type	string	required	The field type for this column.	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/ui/serverWidget Module Script Sample](#).

```
...
var list = serverWidget.createList({
    title : 'Simple List'
});
list.addColumn({
    id : 'column1',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    align : serverWidget.LayoutJustification.RIGHT
});
```

List.addEditColumn(options)

Method Description	Adds a column containing Edit or Edit/View links to a Suitelet or Portlet list. These Edit or Edit/View links appear to the left of a previously existing column.
Returns	serverWidget.ListColumn object
Supported Script Types	Server-side scripts
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

The following code 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 list = serverWidget.createList({
    title : 'Simple List'
});
list.addEditColumn({
    id : 'column1',
    showView : true
});
...
```

List.addPageLink(options)

Method Description	Adds a link to a list.
---------------------------	------------------------

Returns	void
Supported Script Types	Server-side scripts
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

The following code 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 list = serverWidget.createList({
    title : 'Simple List'
});
list.addPageLink({
    title : 'NetSuite',
    type : serverWidget.FormPageLinkType.CROSSLINK,
    url : 'http://www.netsuite.com'
});
```

List.addRow(options)

Method Description	Adds a single row to a list.
Returns	serverWidget.List
Supported Script Types	Server-side scripts
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

The following code 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 list = serverWidget.createList({
    title : 'Simple List'
});
list.addRow({
    row : { columnid1 : 'value1', columnid2 : 'value2' }
});
...
```

List.addRows(options)

Method Description	Adds multiple rows to a list.
Returns	serverWidget.ListColumn
Supported Script Types	Server-side scripts
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	Version 2015 Release 2

Parameter	Type	Required / Optional	Description	Since
			array of name/value pairs. Each pair should contain the value for the corresponding Column object in the list.	

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](#).

```
...
var list = serverWidget.createList({
    title : 'Simple List'
});
list.addRows([
    rows : { columnid1 : 'value1', columnid2 : 'value2' },
    { columnid1 : 'value2', columnid2 : 'value3' }
]);
...
```

List.clientScriptFileDialog

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

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](#).

```
...
var list = serverWidget.createList({
    title : 'Simple List'
});
list.clientScriptFileDialog = 123;
...
```

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

The following code 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 list = serverWidget.createList({
    title : 'Simple List'
});
list.style = serverWidget.ListStyle.REPORT;
...
```

List.title

Property Description	Sets the list title.
Type	string
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var list = serverWidget.createList({
    title : 'Simple List'
});
var title = list.title;
...
```

serverWidget.ListColumn

Object Description	Encapsulates a list column
Supported Script Types	Server-side scripts
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
```

```

var list = serverWidget.createList({
    title : 'Simple List'
});
var listcolumn = list.addColumn({
    id : 'column1',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    align : serverWidget.LayoutJustification.RIGHT
});
...

```

ListColumn.addParamToURL(options)

Method Description	Adds a URL parameter (optionally defined per row) to the list column's URL.
Returns	serverWidget.ListColumn object
Supported Script Types	Server-side scripts
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.param	string	required	The name for the parameter.
options.value	string	required	The value for the parameter.
options.dynamic	boolean	optional	If true, then the parameter value is actually an alias that is calculated per row.

Syntax

The following code 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 list = serverWidget.createList({
    title : 'Simple List'
});
var listcolumn = list.addColumn({
    id : 'column1',
    type : serverWidget.FieldType.URL,
    label : 'URL',
});
listcolumn.addParamToURL({
    param : 'index',
}

```

```
        value : '3'
})
...

```

ListColumn.setURL(options)

Method Description	Sets the base URL for the list column.
Returns	serverWidget.ListColumn
Supported Script Types	Server-side scripts
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
options.dynamic	boolean	optional	If true, then the URL is actually an alias that is calculated per row.

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](#).

```
...
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'
})
...

```

ListColumn.label

Property Description	This list column label.
-----------------------------	-------------------------

Type	string
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var list = serverWidget.createList({
    title : 'Simple List'
});
var listcolumn = list.addColumn({
    id : 'column1',
    type : serverWidget.FieldType.URL,
    label : 'URL',
});
var label = listcolumn.label;
...
```

serverWidget.Sublist

Object Description	Encapsulates a sublist on a <code>serverWidget.Form</code> or an <code>serverWidget.createAssistant(options)</code> object. To add a sublist, use <code>Assistant.addSublist(options)</code> or <code>Form.addSublist(options)</code> . Note: This object is read-only except for instances created via the <code>serverWidget</code> module using Suitelets or <code>beforeLoad</code> user event scripts.
Supported Script Types	Server-side scripts
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
...
```

Sublist.addButton(options)

Method Description	Adds a button to a sublist.
Returns	serverWidget.Button
Supported Script Types	Server-side scripts
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

The following code 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 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'
});
...
```

Sublist.addField(options)

Method Description	Adds a field to a sublist.
Returns	serverWidget.Field object
Supported Script Types	Server-side scripts

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: If you have set the type parameter to SELECT, and you want to add custom options to the select field, you must set source to NULL.	Version 2015 Release 2
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. Note: If you want to add custom options on a select field, you must set the source parameter to NULL.	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/ui/serverWidget Module Script Sample](#).

```
...
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.addMarkAllButtons()

Method Description	Adds a Mark All and an Unmark All button to a LIST type of sublist.
Returns	A serverWidget.Button[] object
Supported Script Types	Server-side scripts
Governance	None
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```

...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
sublist.addMarkAllButtons();
...

```

Sublist.addRefreshButton()

Method Description	Adds a Refresh button to a LIST type of sublist.
Returns	serverWidget.Button object
Supported Script Types	Server-side scripts
Governance	None
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```

...

```

```

var form = serverWidget.createForm({
    title : 'Simple Form'
});
var sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
sublist.addRefreshButton();
...

```

Sublist.getSublistValue(options)

Method Description	Gets a field value on a sublist.
Returns	void
Supported Script Types	Server-side scripts
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 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 Scriptable Sublists .
options.line	number	required	The line number for this field. Note: The first line number on a sublist is 0 (not 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/ui/serverWidget Module Script Sample](#).

```

...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
}

```

```
});
var sublistvalue = sublist.setSublistValue({
    id : 'sublist',
    line: 1
})
...

```

Sublist.setSublistValue(options)

Method Description	Sets the value of a cell in a sublist field.
Returns	void
Supported Script Types	Server-side scripts
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

The following code 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 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"
})
...

```

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	Server-side scripts
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

The following code 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 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.updateTotallingFieldId({
    id : 'fieldId'
})
...

```

Sublist.updateUniqueFieldId(options)

Method Description	Updates a field ID that is to have unique values across the rows in the sublist. Note: This method is available on inlineeditor and editor sublists only.
Returns	serverWidget.Sublist object
Supported Script Types	Server-side scripts

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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
sublist.addField({
    id : 'fieldId',
    type : serverWidget.FieldType.DATE,
    label : 'Date'
});
sublist.updateUniqueId({
    id : 'fieldId'
})
...

```

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

The following code 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 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;
...
```

Sublist.helpText

Property Description	The inline help text for a sublist.
Type	string
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
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.";
...
```

Sublist.label

Property Description	The label for this sublist.
Type	string
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
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;
...
```

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	The count of line items on a sublist as a number.
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
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;
...
```

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)
---------------------------	--

Supported Script Types	Server-side scripts
Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var tab = form.addTab({
    id : 'tabId',
    label : 'Tab'
});
...
```

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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var tab = form.addTab({
    id : 'tabId',
    label : 'Tab'
});
tab.helpText = 'Help Text Goes Here';
...
```

Tab.label

Property Description	The label for a tab or subtab.
Type	string

Module	N/ui/serverWidget 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/ui/serverWidget Module Script Sample](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var tab = form.addTab({
    id : 'tabId',
    label : 'Tab'
});
var label = tab.label;
...
```

serverWidget.createAssistant(options)

Method Description	Creates an assistant object.
Returns	serverWidget.Assistant object
Supported Script Types	Server-side scripts
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 assistant. This title appears at the top of all assistant pages.	Version 2015 Release 2
options.hideNavBar	boolean true false	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

The following code 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'
});
...
```

serverWidget.createForm(options)

Method Description	Creates a form object.
Returns	serverWidget.Form object
Supported Script Types	Server-side scripts
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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
```

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

serverWidget.createList(options)

Method Description	Instantiates a standalone list.
Returns	serverWidget.List object
Supported Script Types	Server-side scripts
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 <code>false</code> . The header appears in the top-right corner on the form. If set to <code>true</code> , the header on the assistant is hidden from view.	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/ui/serverWidget Module Script Sample](#).

```
...  
var list = serverWidget.createList({  
    title : 'Simple List'  
});  
...
```

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 `finish` action is submitted, by default, the text “Congratulations! You have completed the <assistant title>” appears on the finish page.

In a non-sequential process (steps are unordered), `jump` is used to move to the user’s last action.

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

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](#).

```
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
if (assistant.getLastAction() == serverWidget.AssistantSubmitAction.CANCEL) {
    ...
}
```

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. 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

- NONE
- STARTCOL
- STARTROW

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](#).

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

serverWidget.FieldDisplayType

Enum Description	Enumeration that holds the string values for supported field display types. This enum is used to set the value of the displayType parameter when Field.updateDisplayType(options) is called.
Note:	JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateDisplayType({
    displayType = serverWidget.FieldDisplayType.HIDDEN;
});
...
...
```

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. 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
- OUTSIDE BELOW
- OUTSIDE ABOVE
- STARTROW

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](#).

```
...
```

```

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;
});
...

```

serverWidget.FieldType

Enum Description	Enumeration that holds the values for supported field types. This enum is used to set the value of the type parameter when Form.addField(options) 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

<ul style="list-style-type: none"> • CHECKBOX • CURRENCY • DATE • DATETIMETZ • EMAIL • FILE • FLOAT • HELP • INLINEHTML • INTEGER • IMAGE • LABEL 	<ul style="list-style-type: none"> • LONGTEXT • MULTISELECT • PASSPORT • PERCENT • PHONE • SELECT • RADIO • RICHTEXT • TEXT • TEXTAREA • TIMEOFDAY • URL
---	--

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](#).

...

```
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
...

```

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

The following code 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 form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addPageLink({
    type : serverWidget.FormPageLinkType.CROSSLINK,
    title : 'NetSuite',
    url : 'http://www.netsuite.com'
})
...
```

serverWidget.LayoutJustification

Enum Description	Enumeration that holds the string values for supported justification layouts. This enum is used to set the value of the <code>align</code> parameter when <code>List.addColumn(options)</code> is called.
-------------------------	---

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

Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Values

- CENTER
- LEFT
- RIGHT

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](#).

```
...
var list = serverWidget.createList({
    title : 'Simple List'
});
list.addColumn({
    id : 'column1',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    align : serverWidget.LayoutJustification.RIGHT
});
...
...
```

serverWidget.ListStyle

Enum Description	Enumeration that holds the string values for supported list styles. This enum is used to set the value of the <code>List.style</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.
Module	N/ui/serverWidget Module
Since	Version 2015 Release 2

Values

- GRID
- REPORT
- PLAIN
- NORMAL

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](#).

```
...
var list = serverWidget.createList({
    title : 'Simple List'
});
list.style = serverWidget.ListStyle.REPORT;
...
```

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

The following code 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 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;
...
```

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
-------------------------	---

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

- EDITOR
- INLINEEDITOR
- LIST
- STATICLIST

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](#).

```
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
...

```

N/url Module

Use the url module to determine URL navigation paths within NetSuite and format URL strings.

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.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.

N/url Module Script Sample

The following example retrieves the relative url of a record.

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/url'],
    function(url) {
        function resolveRecordUrl() {
            var output = url.resolveRecord({
                recordType: 'salesorder',
                recordId: 6,
                isEditMode: true
            });
            resolveRecordUrl();
        }
    });

```

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

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](#).

```
...
output = url.format(
    'http://fruitland.co',
    {
        fruit: 'grape',
        seedless: true,
        variety: 'concord',
        PLU: 4272
    }
);
...
...
```

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	Version 2015 Release 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'.	Version 2015 Release 1
options.recordId	string	required	The record ID of the target record.	Version 2015 Release 1
options.isEditMode	Boolean	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> .	Version 2015 Release 1
options.params	Object	required	Object used to add parameters for a custom URL. For example, a query to a database or to a search engine	Version 2015 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/url Module Script Sample](#).

```
...
var output = url.resolveRecord({
    recordType: 'salesorder',
    recordId: 6,
    isEditMode: true
    params: p
});
...
```

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	Version 2015 Release 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.	Version 2015 Release 1
options.deploymentId	string	required	The internal ID of the deployment script	Version 2015 Release 1
options.returnExternalUrl	Boolean	required	Indicates whether to return the External URL.	Version 2015 Release 1
options.params	Object	required	The object containing name/value pairs to describe the query.	Version 2015 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/url Module Script Sample](#).

```
...
```

```
var output = url.resolveScript({
   scriptId: 'custom_script',
   deploymentId: 'custom_script_deployment',
   returnExternalUrl: true
    params: p
});
...
```

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	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 for the tasklink. Note: Each page in NetSuite has a unique Tasklink Id associated with it for a given record type. You can determine the Tasklink for a page within NetSuite by viewing the HTML page source. Search for a string similar to the following, where LIST_SCRIPT refers to the TASKLINK: onclick="nlPopupHelp('LIST_SCRIPT','help')."	Version 2015 Release 1
options.params	Map	optional	The Map object containing name/value pairs to describe the query.	Version 2015 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/url Module Script Sample](#).

```
...
u = url.resolveTaskLink('SRCH_JOB', p);
...
```

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	<code>util.isArray(obj)</code>	boolean true false	Returns true if the <code>obj</code> parameter is a JavaScript and false otherwise.
	<code>util.isBoolean(obj)</code>	boolean true false	Returns true if the <code>obj</code> parameter is a Boolean and false otherwise.
	<code>util.isDate(obj)</code>	boolean true false	Returns true if the <code>obj</code> parameter is a JavaScript Date object and false otherwise.
	<code>util.each(iterable, callback)</code>	Object or Array	Iterates over each member in an Object or Array.
	<code>util.extend(receiver, contributor)</code>	Object	Copies the properties in a source object to a destination object and returns the destination object.
	<code>utilisFunction(obj)</code>	boolean true false	Returns true if the <code>obj</code> parameter is a JavaScript Function object and false otherwise.
	<code>util.isNumber(obj)</code>	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.
	<code>util.isobject(obj)</code>	boolean true false	Returns true if the <code>obj</code> parameter is a strictly a JavaScript Object, and false otherwise.
	<code>util.isRegExp(obj)</code>	boolean true false	Returns true if the <code>obj</code> parameter is a JavaScript RegExp object or a value

Member Type	Name	Return Type / Value Type	Description
			that evaluates to a RegExp object, and false otherwise.
	util.isString(obj)	boolean true false	Returns true if the obj 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 obj 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

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

```
...
var records = ["Sales Order", "Invoice", "Item Fulfillment"];
util.isArray(records); // returns true

var record = "Sales Order";
util.isArray(record); // returns false
...
```

util.isBoolean(obj)

Method Description	Returns true if the obj 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
obj	Object Primitive	Required	Object for which you want to verify the type.	Version 2016 Release 1

Syntax

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

```
...
var flag = true;
util.isBoolean(flag); // returns true
util.Boolean(true); // returns true

util.Boolean(1); // returns false
...
```

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

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

```
...
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
...
```

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
<code>obj</code>	Object Primitive	Required	Object for which you want to verify the type.	Version 2016 Release 1

Syntax

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

```
...
function test() {};
var test2 = function() {};

util.isFunction(test); // returns true
util.isFunction(test2); // returns true
...
```

util.isNumber(obj)

Method Description	Returns true if the <code>obj</code> 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
<code>obj</code>	Object Primitive	Required	Object for which you want to verify the type.	Version 2016 Release 1

Syntax

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

```
...
util.isNumber(112); // returns true
util.isNumber("112"); // returns false
util.isNumber(NaN); // returns true

var testNum = 112;
util.isNumber(testNum.valueOf()); // returns true
...
```

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
obj	Object Primitive	Required	Object for which you want to verify the type.	Version 2016 Release 1

Syntax

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

```
...
util.isobject({});           // returns true
util.isobject(function() {}); // returns false
...
```

util.isRegExp(obj)

Method Description	Returns true if the obj 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

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

```
...
```

```
util.isRegExp(/this is a regexp/);           // returns true
util.isRegExp(new RegExp('this is another regexp')); // returns true
...

```

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
<code>obj</code>	Object Primitive	Required	Object for which you want to verify the type.	Version 2016 Release 1

Syntax

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

```
...
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
...
```

util.nanoTime()

Method Description	Returns the current time (epoch) in nanoseconds. You can use this method to measure elapsed time between two events. For example, the following code sample shows how to calculate the number of nanoseconds between two call to <code>util.nanoTime()</code> :
	<pre>var startTime = util.nanoTime(); ... var elapsedTime = util.nanoTime() - startTime;</pre>

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

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

```
...
util.isString('');
util.isString('a string');           // returns true
util.isString('another string');
util.isString(myString);            // returns true
util.isString(null);                // returns false
...
```

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 <code>iterable</code> .
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

Parameter	Type	Required / Optional	Description	Since
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

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

```
...
// 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');
});
...
...
```

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

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:

```
...
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);
});

...

```

The following snippet shows overriding two objects with a few similar keys:

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

```

N/workflow Module

This module loads the workflow module to initiate new workflow instances or trigger existing workflow instances.

N/workflow Module Members

Member Type	Name	Return Type / Value Type	Description
Method	workflow.initiate(options)	number	<p>Initiates a workflow on-demand. This method is the programmatic equivalent of the Initiate Workflow Action action in SuiteFlow.</p> <p>Returns the internal ID (number) of the workflow instance used to track the workflow against the record.</p>

Member Type	Name	Return Type / Value Type	Description
	workflow.trigger(options)	number	<p>Triggers a workflow on a record. The actions and transitions of the workflow are evaluated for the record in the workflow instance, based on the current state for the workflow instance.</p> <p>Returns the internal ID (number) of the workflow instance used to track the workflow against the record.</p>

N/workflow Module Script Sample

The following example searches for a specific workflow deployed on the customer record and then executes it.

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/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	<p>Initiates a workflow on-demand. This method is the programmatic equivalent of the Initiate Workflow Action action in SuiteFlow.</p> <p>Returns the internal ID of the workflow instance used to track the workflow against the record.</p>
---------------------------	---

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.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

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](#).

```
...
var workflowInstanceId = workflow.initiate({
    recordType: 'customer',
    recordId: 24,
    workflowId: 'customworkflow_myWorkflow'
});
...
```

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

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](#).

```
...
var workflowInstanceId = workflow.trigger({
    recordType: 'salesorder',
    recordId: 1234,
    workflowId: 'custworkflow_foobar',
    defaultValues: p
    actionId: workflowaction25
});
...
```

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

Member Type	Name	Return Type / Value Type	Description
			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.
	<code>xml.Element</code>	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 <code>TEXT_NODE</code> .
	<code>xml.Attr</code>	Object	Represents an attribute node of an <code>xml.Element</code> object.
Method	<code>xml.escape(options)</code>	string	Prepares a string for use in XML by escaping XML markup, such as angle brackets, quotation marks, and ampersands.
	<code>xml.validate(options)</code>	void	Validates an XML document against an XML Schema (XSD).
Enum	<code>xml.NodeType</code>	string (read-only)	Enumeration that holds the string values for the supported node types. The <code>Node.nodeType</code> 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	<code>Parser.fromString(options)</code>	<code>xml.Document</code>	Parses a string into a W3C XML document object.
	<code>Parser.toString(options)</code>	string	Converts (serializes) an <code>xml.Document</code> 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	<code>XPath.select(options)</code>	<code>xml.Node[]</code>	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.
	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).

Member Type	Name	Return Type / Value Type	Description
	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 xml.Element 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 xml.Element , the name is the name of the element.
	Node.nodeType	string	The type of node defined as a value from the xml.NodeType 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 xml.Document 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.

Member Type	Name	Return Type / Value Type	Description
	Node.previousSibling	xml.Node (read-only)	The previous node in a node list or null if the current node is the first node.
	Node.textContent	string	The textual content of a node and its descendants.

Document Object Members

The following members are called on the `xml.Document` object.

Member Type	Name	Return Type / Value Type	Description
Method	Document.adoptNode(options)	xml.Node	Attempts to adopt a node from another document to this document.
	Document.createAttribute(options)	xml.Attr	Creates an attribute node of type ATTRIBUTE_NODE with the optional specified value.
	Document.createAttributeNS(options)	xml.Attr	Creates an attribute node of type ATTRIBUTE_NODE, with the specified namespace value and optional specified value.
	Document.createCDATASection(options)	xml.Node	Creates a CDATA section node of type DOCUMENT_FRAGMENT_NODE with the specified data.
	Document.createComment(options)	xml.Node	Creates a Comment node of type COMMENT_NODE with the specified string.
	Document.createDocumentFragment()	xml.Node	Creates a node of type DOCUMENT_FRAGMENT_NODE.
	Document.createElement(options)	xml.Element	Creates a new node of type ELEMENT_NODE with the specified name.
	Document.createElementNS(options)	xml.Element	Creates a new node of type ELEMENT_NODE with the specified namespace URI and name.
	Document.createProcessingInstruction(options)	xml.Node	Creates a new node of type PROCESSING_INSTRUCTION_NODE with the specified target and data.
	Document.createTextNode(options)	xml.Node	Creates a new node of type TEXT_NODE.
	Document.getElementById(options)	xml.Element	Returns the element that has an ID attribute with the specified value as an <code>xml.Element</code> object.
	Document.getElementsByTagName(options)	xml.Element[]	Returns an array of <code>xml.Element</code> objects with a specific tag name, in the

Member Type	Name	Return Type / Value Type	Description
			order in which they appear in the XML document.
	Document.getElementsByTagNameNS(options)	xml.Element[]	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.
	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 <code>null</code> 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 <code>true</code> if the current XML document is standalone or returns <code>false</code> if it is not.
	Document.xmlVersion	string	Part of the XML declaration, the version number of the XML document.

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](#).

Element 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 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

Member Type	Name	Return Type / Value Type	Description
			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 <code>xml.Element</code> object.

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](#).

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)	Name of an attribute.
	Attr.ownerElement	<code>xml.Element</code> (read-only)	The <code>xml.Element</code> object that is the parent of the <code>xml.Attr</code> 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

TBD

xml.Parser

Object Description	Encapsulates the functionality used by NetSuite to parse an XML document.
Supported Script Types	All script types
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
define([ 'N/xml' ], function(xml) {
    ...
})
```

```
var parserObj = xml.Parser;
...
```

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 options.text argument, Parser.fromString returns an SSS_XML_DOM_EXCEPTION error.
Returns	xml.Document
Supported Script Types	All script types
Governance	None
Module	N/xml Module
Since	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

```
define(['N/xml', 'N/file'], function(xml, file) {
    ...
    var xmlStringContent = file.load('BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });
    ...
});
```

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

```
define([ 'N/xml' ], function(xml) {
    ...
    var xmlStringContent = xml.Parser.toString({
        document : xmlFileContent
    });
    ...
})
```

xml.XPath

Object Description	Encapsulates the functionality to run XPath expressions.
Supported Script Types	All script types
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
define([ 'N/xml' ], function(xml) {
    ...
    var xpath = xml.XPath;
    ...
})
```

XPath.select(options)

Method Description	Selects an array of nodes from an XML that match an XPath expression.
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.node	xml.Node	Required	XML node being queried.
options.xpath	string	Required	XPath expression used to query node.

Syntax

```
define([ 'N/xml' ], function(xml) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var bookNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });
    ...
});
```

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.
Note:	NetSuite supports a subset of W3C DOM methods. See Node Object Members .
Supported Script Types	All script types
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
define([ 'N/xml' ], function(xml) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var bookNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var nodeValue1 = bookNode[0].firstChild.nextSibling.textContent;
```

```
var nodeValue2 = bookNode[0].getElementsByTagName({
    tagName : 'author'
})[0].textContent;
```

...

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

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

```
define(['N/xml', 'N/file'], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagName({
        tagName : 'author'
    });

    var node = parentNode[1].firstChild.nextSibling;

    var newnode = elem[0].appendChild({
        newChild : node
    });
});
```

...

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

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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDoc = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDoc,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagName({
        tagName : 'author'
    });

    var copiednode = elem[0].cloneNode({
        deep : true
    });
    ...
});
```

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.
---------------------------	--

	<ul style="list-style-type: none"> 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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagName({
        tagName : 'author'
    });
});
```

```
var posCode = elem[0].compareDocumentPosition({
    other : elem[1]
});
...
```

Node.hasAttributes()

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.
	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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var hasAttributes = parentNode[0].hasAttributes()
    ...
}
```

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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
```

```

...
var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
var xmlDocument = xml.Parser.fromString({
    text : xmlFileContent
});

var parentNode = xml.XPath.select({
    node : xmlDocument,
    xpath : '//book'
});

var hasChildren = parentNode[0].hasChildNodes()
...

```

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 refChild is , the method inserts the new node at the end of the list of children.

Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	HIERARCHY_REQUEST_ERR: An attempt was made to insert a node where it is not permitted.	Node cannot be inserted.

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({

```

```

        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var elemList1 = parentNode[0].getElementsByTagName({
        tagName : 'author'
    });
    var elemList2 = parentNode[1].getElementsByTagName({
        tagName : 'author'
    });

    var insertedNode = parentNode[0].insertBefore({
        newChild : elemList1[0],
        refChild : elemList2[0]
    });
    ...

```

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 . 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

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
<code>options.namespaceURI</code>	<code>string</code>	Required	The namespace URI to compare.

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

```

```

var parentNode = xml.XPath.select({
    node : xmlDocument,
    xpath : '//book'
});

var isDefault = parentNode[0].isDefaultNamespace({
    namespaceURI : '*'
});
...

```

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. Note: Two nodes may be equal, even if they are not the same. See Node.isSameNode(options) . Important: This method is not supported on Internet Explorer.
Returns	<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

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

```

```

var parentNode = xml.XPath.select({
    node : xmlDocument,
    xpath : '//book'
});

var elem = parentNode[0].getElementsByTagName({
    tagName : 'title'
});
var node = parentNode[0].firstChild.nextSibling;

var isEqual = elem[0].isEqual({
    other : node
});
...

```

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. Note: Two nodes that are the same are also equal. See Node.isEqualNode(options) . Important: This method is not supported on Internet Explorer or Firefox.
Returns	<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>	xml.Node	Required	The node to compare with the current node.

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
}

```

```

});  
  

var elem = parentNode[0].getElementsByTagName({  

    tagName : 'title'  

});  

var node = parentNode[0].firstChild.nextSibling;  
  

var isSame = elem[0].isSameNode({  

    other : node  

});  

...

```

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	<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.prefix</code>	<code>string</code>	Required	Namespace prefix associated with the namespace URI.

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var uri = parentNode[0].lookupNamespaceURI({
        prefix : '*'
    });
}

```

...

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 the namespace prefix.

Syntax

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDoc = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDoc,
        xpath : '//book'
    });

    var prefix = parentNode[0].lookupPrefix({
        namespaceURI : '*'
    });
    ...
});
```

Node.normalize()

Method Description	Puts all text nodes underneath a node, including attribute nodes, into a normal form. In normal form, only structure, for example, elements, comments, processing instructions,
---------------------------	---

	CDATA sections, and entity references, separate 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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var node = parentNode[0].firstChild.nextSibling;
    node.normalize();
    ...
});
```

Node.removeChild(options)

Method Description	Removes the specified child node. the removed 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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var node = parentNode[0].firstChild.nextSibling;

    var removednode = parentNode[0].removeChild({
        oldChild : node
    });
    ...
});
```

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.
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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagName({
        tagName : 'title'
    });
    var node = parentNode[0].firstChild.nextSibling;

    var replacednode = parentNode[0].replaceChild({
        newChild : elem[0],
        oldChild : node
    });
    ...
});
```

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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
```

```

});  
  

var elem = parentNode[0].getElementsByTagName({  

    tagName : 'title'  

});  
  

var attrs = elem[0].attributes;  

...

```

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	<code>string</code> (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {  

    ...  

    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();  

    var xmlDoc = xml.Parser.fromString({  

        text : xmlFileContent  

    });  
  

    var parentNode = xml.XPath.select({  

        node : xmlDoc,  

        xpath : '//book'  

    });  
  

    var baseuri = parentNode[0].baseURI;  

    ...
}

```

Node.childNodes

Property Description	Array of all child nodes of a node or an empty array if there are no child nodes.
Type	<code>xml.Node[]</code>
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {  

    ...  

    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();  

    var xmlDoc = xml.Parser.fromString({
}

```

```

        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var childnodes = parentNode[0].childNodes;
    ...

```

Node.firstChild

Property Description	First child node for a specific text or element node, or <code>null</code> if there are no child nodes.
Type	<code>xml.Node</code>
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml' ], function(xml) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString(xmlFileContent);

    var bookNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var nodeValue1 = bookNode[0].firstChild.nextSibling.textContent;
    ...

```

Node.lastChild

Property Description	Last child node for a specific element or text node, or <code>null</code> if there are no child nodes.
Type	<code>xml.Node</code>
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

```

```

var parentNode = xml.XPath.select({
    node : xmlDocument,
    xpath : '//book'
});

var nodeValue = parentNode[0].lastChild.previousSibling.textContent;
...

```

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

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var localname = parentNode[0].localName;
    ...
}

```

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

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        ...
    });
}

```

```

        xpath : '//book'
});

var uri = parentNode[0].namespaceURI;
...

```

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

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var nodeName = parentNode[0].firstChild.nextSibling.textContent;
    ...
}

```

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

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        ...
}

```

```

        xpath : '//book'
    });

var nodeName = parentNode[0].firstChild.nodeName;
...

```

Node.nodeType

Property Description	The type of node as an enum. For all possible values of this property, see xml.NodeType .
Type	xml.NodeType
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var nodeType = parentNode[0].firstChild.nodeType;
    ...
}

```

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	<code>string</code>
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        ...
}

```

```

        xpath : '//book'
    });

var nodeValue = parentNode[0].firstChild.nodeValue;
...

```

Node.ownerDocument

Property Description	The root element for a node as a xml.Document object. Use this object used to create new nodes with Document.createElement(options) or Document.createElementNS(options) .
Type	xml.Document
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var doc = parentNode[0].ownerDocument;
    ...
}

```

Node.parentNode

Property Description	The parent node of a node. All node types, except xml.Attr , xml.Document , DocumentFragment , Entity , and Notation can have a parent node. See xml.NodeType for possible node types.
Type	xml.Node
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });
}

```

```

var parentNode = xml.XPath.select({
    node : xmlDocument,
    xpath : '//book'
});

var nodevalue = parentNode[0].lastChild.parentNode.textContent;
...

```

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	string
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

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var namespacePrefix = parentNode[0].firstChild.prefix;
    ...
}

```

Node.previousSibling

Property Description	The previous node in a node list or <code>null</code> if the current node is the first node.
Type	xml.Node
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {

```

```

...
var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
var xmlDocument = xml.Parser.fromString({
    text : xmlFileContent
});

var parentNode = xml.XPath.select({
    node : xmlDocument,
    xpath : '//book'
});

varnodeValue = parentNode[0].lastChild.previousSibling.textContent;
...

```

Node.textContent

Property Description	The textual content of a node and its descendants. If the value is null, 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	string
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    varnodeValue = parentNode[0].firstChild.textContent;
    ...
}

```

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. Note: An XML document object is also a node, of type DOCUMENT_NODE. As a result, you can use the methods for the <code>xml.Node</code> object with the Document object.
Supported Script Types	All script types
Module	N/xml Module

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

Syntax

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });
    ...
});
```

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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample1.xml').getContents();
```

```

var xmlDocument1 = xml.Parser.fromString({
    text : xmlFile1Content
});

var xmlFile2Content = file.load('SuiteScripts/BookSample2.xml').getContents();
var xmlDocument2 = xml.Parser.fromString({
    text : xmlFile2Content
});
var sourceNode = xml.XPath.select({
    node : xmlDocument2,
    xpath : '//book'
});

var adoptedNode = xmlDocument1.adoptNode({
    source : sourceNode,
});
...

```

Document.createAttribute(options)

Method Description	Creates an attribute node of type ATTRIBUTE_NODE with the optional specified value and returns the new xml.Attr object. The localName, prefix, and namespaceURI properties of the new node are set to <code>null</code> .
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 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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
```

```

...
var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
var xmlDocument = xml.Parser.fromString({
    text : xmlFileContent
});

var attr = xmlDocument.createAttribute({
    name : 'lang',
    value : 'fr'
});
...

```

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 xml.Attr object. The Node.localName , Node.prefix , and Node.namespaceURI properties of the new node are set to <code>null</code> . 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.namespaceURI	string	Required	Namespace URI of the attribute to create. Value can be <code>null</code> .
options.qualifiedName	string	Required	Qualified name of the new attribute node.
options.value	string	Optional	Value for the attribute node. If unspecified, the value is an empty string.

Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.	Attribute with the specified value cannot be created.

Syntax

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var attr = xmlDocument.createAttributeNS({
        namespaceURI : '*',
        qualifiedName : 'lang',
        value : 'fr'
    });
    ...
});
```

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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });
});
```

```
var newNode = xmlDocument.createCDATASection({
    data : 'Limited Edition.'
});
...

```

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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var newNode = xmlDocument.createComment({
        data : 'This is a comment.'
    });
    ...
}
```

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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var newNode = xmlDocument.createDocumentFragment();
    ...
});
```

Document.createElement(options)

Method Description	Creates a new node of type ELEMENT_NODE with the specified name and returns the new xml.Element node. The Node.localName , Node.prefix , and Node.namespaceURI properties of the new node are set to <code>null</code> .
Returns	xml.Element
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	Name of the element to create.

Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.	Element cannot be created with the specified tagName value.

Syntax

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
});
```

```
});

var elem = xmlDocument.createElement({
    tagName : 'book'
});
...
}
```

Document.createElementNS(options)

Method Description	Creates a new node of type ELEMENT_NODE with the specified namespace URI and name and returns the new xml.Element object. The Node.localName , Node.prefix , and Node.namespaceURI properties of the new node are set to <code>null</code> .
Returns	xml.Element
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 element to create. Can be <code>null</code> .
options.qualifiedName	string	Required	Qualified name of the element to create.

Errors

Error Code	Message	Thrown If
SSS_XML_DOM_EXCEPTION	INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.	Element with the specified namespace cannot be created.

Syntax

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });
}
}
```

```
var elem = xmlDocument.createElementNS({
    namespaceURI : '*',
    qualifiedName : 'book'
});
...

```

Document.createProcessingInstruction(options)

Method Description	Creates a new node of type PROCESSING_INSTRUCTION_NODE with the specified target and data and returns the new xml.Node object. The following example shows a sample processing instruction: <?xml version="1.0"?> 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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });
}
```

```

var newNode = xmlDocument.createProcessingInstruction({
    target : 'xml'
    data : 'version="1.0"'
});
...

```

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

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
options.data	string	Required	Data for the text node.

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var newNode = xmlDocument.createTextNode({
        data : 'Sample Title'
    });
    ...
}

```

Document.getElementById(options)

Method Description	Returns the element that has an ID attribute with the specified value as an xml.Element object. Returns <code>null</code> if no such element exists.
Returns	xml.Element
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.elementId	string	Required	Unique ID value for an element.

Syntax

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var elem = xmlDocument.getElementById({
        elementId : 'id12345'
    });
    ...
})
```

Document.getElementsByTagName(options)

Method Description	Returns an array of xml.Element objects with a specific tag name, in the order in which they appear in the XML document.
Returns	xml.Element[]
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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
```

```

...
var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
var xmlDocument = xml.Parser.fromString({
    text : xmlFileContent
});

var elem = xmlDocument.getElementsByTagName({
    tagName : 'book'
});
...

```

Document.getElementsByTagNameNS(options)

Method Description	Returns an array of xml.Element objects with a specific tag name and namespace, in the order in which they appear in the XML document.
	Important: This method is not supported on Internet Explorer.
Returns	xml.Element[]
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

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var elem = xmlDocument.getElementsByTagNameNS({
        namespaceURI : '*',
        localName : 'book'
    });
    ...
}

```

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 <code>xml.Node</code> object.
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
<code>options.importedNode</code>	<code>xml.Node</code>	Required	Node from another XML document to import.
<code>options.deep</code>	<code>boolean true false</code>	Required	Use <code>true</code> to import the node, its attributes, and all descendants. Use <code>false</code> to only import the node and its attributes. Important: This parameter is not supported on Internet Explorer.

Errors

Error Code	Message	Thrown If
<code>SSS_XML_DOM_EXCEPTION</code>	<code>NOT_SUPPORTED_ERR</code> : The implementation does not support the requested type of object or operation.	Node cannot be imported.

Syntax

```
define(['N/xml', 'N/file'], function(xml, file) {
    ...
    var xmlFile1Content = file.load('SuiteScripts/BookSample1.xml').getContents();
    var xmlDocument1 = xml.Parser.fromString({
        text : xmlFile1Content
    });

    var xmlFile2Content = file.load('SuiteScripts/BookSample2.xml').getContents();
    var xmlDocument2 = xml.Parser.fromString({
```

```

        text : xmlFile2Content
    });
    var foreignNode = xml.XPath.select({
        node : xmlDocument2,
        xpath : '//book'
    });

    var importedNode = xmlDocument1.importNode({
        importedNode : foreignNode,
        deep : true
    });
    ...

```

Documentdoctype

Property Description	The doctype of the XML document.
	Important: This property is not supported on Internet Explorer.
Type	xml.Element (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var doctype = xmlDocument.doctype;
    ...

```

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	xml.Element (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({

```

```

        text : xmlFileContent
});

var root = xmlDocument.documentElement;
...

```

Document.documentElement

Property Description	Location of the document or <code>null</code> if undefined. For example, the <code>documentURI</code> of the <code>BookSample.xml</code> sample is <code>BookSample.xml</code> . See the help topic <i>books.xml Sample XML File</i> .
Type	<code>string</code>
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var documentURI = xmlDocument.documentElement;
    ...
}

```

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 <code>UTF-8</code> : <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
Type	<code>string</code> (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });
}

```

```
var encoding = xmlDocument.xmlEncoding;
...
```

Document.xmlEncoding

Property Description	Part of the XML declaration, the XML encoding of the XML document. In the following declaration, the <code>xmlEncoding</code> property is UTF-8: <code><?xml version="1.0" encoding="UTF-8" standalone="yes"?></code>
Type	string (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var encoding = xmlDocument.xmlEncoding;
    ...
}
```

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 <code>xmlStandalone</code> property is <code>true</code> : <code><?xml version="1.0" encoding="UTF-8" standalone="yes"?></code>
Type	boolean <code>true</code> <code>false</code>
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });
    ...
}
```

```
});
var isStandalone = xmlDocument.xmlStandalone;
...
```

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

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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var version = xmlDocument.xmlVersion;
    ...
}
```

xml.Element

Object Description	Represents an element in an XML document. Elements may contain attributes, other elements, or text. If an element contains text, the text is represented in a text node of type TEXT_NODE. For example, the following element year contains a text node with the value of 2015: <year>2015</year> Note: An XML element object is also a node, of type ELEMENT_NODE. As a result, you can use the methods for the xml.Node object with the Element object.
Supported Script Types	All script types

Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagName({tagName : 'title'});
    ...
});
```

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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var elem = parentNode[0].getAttribute({name : 'title'});
    ...
});
```

```

});  
  

var elem = parentNode[0].getElementsByTagName({tagName : 'title'});  

var attr = elem[0].getAttribute({  

    name : 'lang'  

});  

...

```

Element.getAttributeNode(options)

Method Description	Returns an attribute node by name or null if there is no attribute.
	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

```

...
var attr = elem[0].getAttributeNode({  

    name : 'lang'  

});  

...

```

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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDoc = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDoc,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagName({tagName : 'title'});
    var attr = elem[0].getAttributeNodeNS({
        namespaceURI : '*'
        localName : 'lang'
    });
    ...
});
```

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

```
define(['N/xml', 'N/file'], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagName({tagName : 'title'});
    var attr = elem[0].getAttributeNS({
        namespaceURI : '*'
        localName : 'lang'
    });
    ...
});
```

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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagName({tagName : 'title'});
    ...
});
```

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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagNameNS({
        namespaceURI : '*',
        localName : 'lang'
    });
    ...
});
```

Element.hasAttribute(options)

Method Description	Returns true if the current element has an attribute with the specified name or if that attribute has a default value. Otherwise, returns false.
Important:	This method is not supported on Internet Explorer.
Returns	boolean true false
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 match on.

Syntax

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
});
```

```

var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
var xmlDoc = xml.Parser.fromString({
    text : xmlFileContent
});

var parentNode = xml.XPath.select({
    node : xmlDoc,
    xpath : '//book'
});

var elem = parentNode[0].getElementsByTagName({tagName : 'title'});

var attrExists = elem[0].hasAttribute({
    name : 'lang'
});
...

```

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

Note: The options parameter is a JavaScript object.

Parameter	Type	Required / Optional	Description
<code>options.namespaceURI</code>	<code>string</code>	Required	Namespace URI of the attribute to match on.
<code>options.localName</code>	<code>string</code>	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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
```

```

...
var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
var xmlDocument = xml.Parser.fromString({
    text : xmlFileContent
});

var parentNode = xml.XPath.select({
    node : xmlDocument,
    xpath : '//book'
});

var elem = parentNode[0].getElementsByTagName({tagName : 'title'});

var attrExists = elem[0].hasAttributeNS({
    namespaceURI : '*',
    localName : 'lang'
});
...

```

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

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({

```

```

        node : xmlDocument,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagName({tagName : 'title'});

    elem[0].removeAttribute({
        name : 'lang'
    });
    ...

```

Element.removeAttributeNode(options)

Method Description	Removes the specified attribute node as a xml Attr object. Returns the removed attribute node.
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

```

...
...
```

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.
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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagName({tagName : 'title'});

    elem[0].removeAttributeNS({
        namespaceURI : '*',
        localName : 'lang'
    });
    ...
});
```

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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagName({tagName : 'title'});

    elem[0].setAttribute({
        name : 'lang',
        value : 'fr'
    });
    ...
});
```

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 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.newAttr	xml Attr	Required	New xml Attr object to add to the xml Element 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

```
...
...
```

Element.setAttributeNodeNS(options)

Method Description	Adds the specified attribute node. If an attribute with the same local name and namespace URL is already present in the element, it is replaced by the new one. If an attribute with the same namespaceURI and localName property already exist, it is replaced with the object in the newAttr parameter. If the attribute node replaces an existing attribute node, the method returns the new xml Attr object. Important: This method is not supported on Internet Explorer.
Returns	xml Attr
Supported Script Types	All script types
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.newAttr	xml.Attr	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

```
...
...
```

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 <code>xml.Attr</code> object. 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 add.
options.qualifiedName	string	Required	Fully qualified attribute name to add.

Parameter	Type	Required / Optional	Description
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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagName({tagName : 'title'});

    elem[0].setAttributeNS({
        namespaceURI : '*',
        qualifiedName : 'lang',
        value : 'fr'
    });
    ...
});
```

Element.tagName

Property Description	The tag name of this <code>xml.Element</code> object.
Type	string (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
```

```

        node : xmlDocument,
        xpath : '//book'
    });

var elem = parentNode[0].getElementsByTagName({tagName : 'title'});
var tagName = elem[0].tagName; \\ returns 'title'.
...

```

xml.Attr

Object Description	Represents an attribute node of an xml.Element object.
Supported Script Types	All script types
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagName({tagName : 'title'});
    var attr = elem[0].getAttributeNode({
        name : 'lang'
    });

    ...
}

```

Attr.name

Property Description	Name of an attribute. If the Node.localName property for the parent xml.Element object is not , this property is a qualified name.
Type	string (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
}

```

```

var xmlDocument = xml.Parser.fromString({
    text : xmlFileContent
});

var parentNode = xml.XPath.select({
    node : xmlDocument,
    xpath : '//book'
});

var elem = parentNode[0].getElementsByTagName({tagName : 'title'});
var attr = elem[0].getAttributeNode({
    name : 'lang'
});

var attrName = attr.name; \\ returns 'lang'.
...

```

Attr.ownerElement

Property Description	xml.Element 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.
	Important: This property is not supported on Internet Explorer.
Type	xml.Element (read-only)
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagName({tagName : 'title'});
    var attr = elem[0].getAttributeNode({
        name : 'lang'
    });

    var attrElement = attr.ownerElement; \\ returns the title element.
    ...

```

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	boolean true false
Module	N/xml Module
Since	Version 2015 Release 2

Syntax

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var parentNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//book'
    });

    var elem = parentNode[0].getElementsByTagName({tagName : 'title'});
    var attr = elem[0].getAttributeNode({
        name : 'lang'
    });

    var attrSpecified = attr.specified;
    ...
});
```

Attr.value

Property Description	<p>Value of an attribute. The value of the attribute is returned as a string.</p> <p>Character and general entity references are replaced with their values. For example, a character reference such as &#160; or an entity reference such as &nbsp; is replaced with a non-breaking space.</p> <p>Note: If you set this value, it creates a text node with the unparsed contents of the string, for example, any characters that an XML processor would recognize as markup are instead treated as literal text.</p>
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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
});
```

```

var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
var xmlDocument = xml.Parser.fromString({
    text : xmlFileContent
});

var parentNode = xml.XPath.select({
    node : xmlDocument,
    xpath : '//book'
});

var elem = parentNode[0].getElementsByTagName({tagName : 'title'});
var attr = elem[0].getAttributeNode({
    name : 'lang'
});

var attrValue = attr.value;
...

```

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

```

define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlEscapedDocument = xml.escape({
        xmlText : xmlFileContent
    });
    ...
}

```

xml.validate(options)

Method Description	Validates an XML document against an XML Schema (XSD).
---------------------------	--

<p>Important: This method only validates XML Schema (XSD); validation of other XML schema languages is not supported.</p> <p>The XML document must be passed as an xml.Document object. The location of the source XML Document does not matter; the validation is performed with the Document object stored in memory. The XSD must be stored in the File Cabinet.</p>	
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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });
});
```

```
xml.validate({
    xml : xmlDocument,
    xsdFilePathOrId : 'SuiteScripts/schema_parent.xsd',
    importFolderPathOrId : 'SuiteScripts/'
});
...
```

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

```
define([ 'N/xml', 'N/file' ], function(xml, file) {
    ...
    var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents();
    var xmlDocument = xml.Parser.fromString({
        text : xmlFileContent
    });

    var DocType = xmlDocument.nodeType; \\ returns DOCUMENT_NODE
    ...
}
```

Chapter 5 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 `*/` in order 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
@NApiVersion	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.
@NModuleScope	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 SameAccount, access to the module is limited to other modules from the same bundle, and modules native to the same source account or sandbox environment. <p>Source code is not hidden at runtime.</p> <ul style="list-style-type: none"> If the value is set to TargetAccount, 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. <p>Source code is hidden at runtime.</p> <ul style="list-style-type: none"> If the value is set to Public, any script in the account can load and use the module. <p>Source code is hidden at runtime.</p> <p>The default value is SameAccount.</p> <p>For more information, see Controlling Access to Scripts and Custom Modules.</p>
@NScriptType	BundleInstallationScript ClientScript MapReduceScript MassUpdateScript Portlet Restlet ScheduledScript Suitelet	Required for entry point scripts	This tag identifies the type of script defined in the file.

JSDoc Tag	Possible Values	Required/Optional	Description
	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	<p>Limits script access to modules native to the same environment in which the script was created and uploaded.</p> <p>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</p>	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
	<p>topic Understanding NetSuite Environments.</p> <p>A script file that is ‘native’ to the environment is added to an account using an authenticated user session.</p>			
TargetAccount	<p>Limits script access to modules native to the same source or target environment as the script.</p> <p>A source environment includes the NetSuite account (and any associated sandboxes) from which a bundle is installed into another account.</p> <p>A target environment includes the NetSuite account (and any associated sandboxes) into which a bundle is installed (whether via Bundle Copy or Bundle Install).</p>	<p>A bundled module that is not native to the target or source account attempts to import the module.</p>	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	<p>Any bundle (native and third party) that is installed in the account can run the script.</p>	<p>The script is not installed in the account.</p>	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

ModuleScope Value	Description	Access Disallowed When:	Visibility of Source Code	Examples
				<p>development account or a Test Drive account to a sandbox or production account.</p> <ul style="list-style-type: none"> • Developing open source code.

For more information about packaging and distributing bundles (SuiteApps), see the help topic [SuiteBundler](#).

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 Environment](#), [NetSuite Development Accounts](#), and the [SuiteApp Development Process with SuiteBundler](#).

Chapter 6 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.

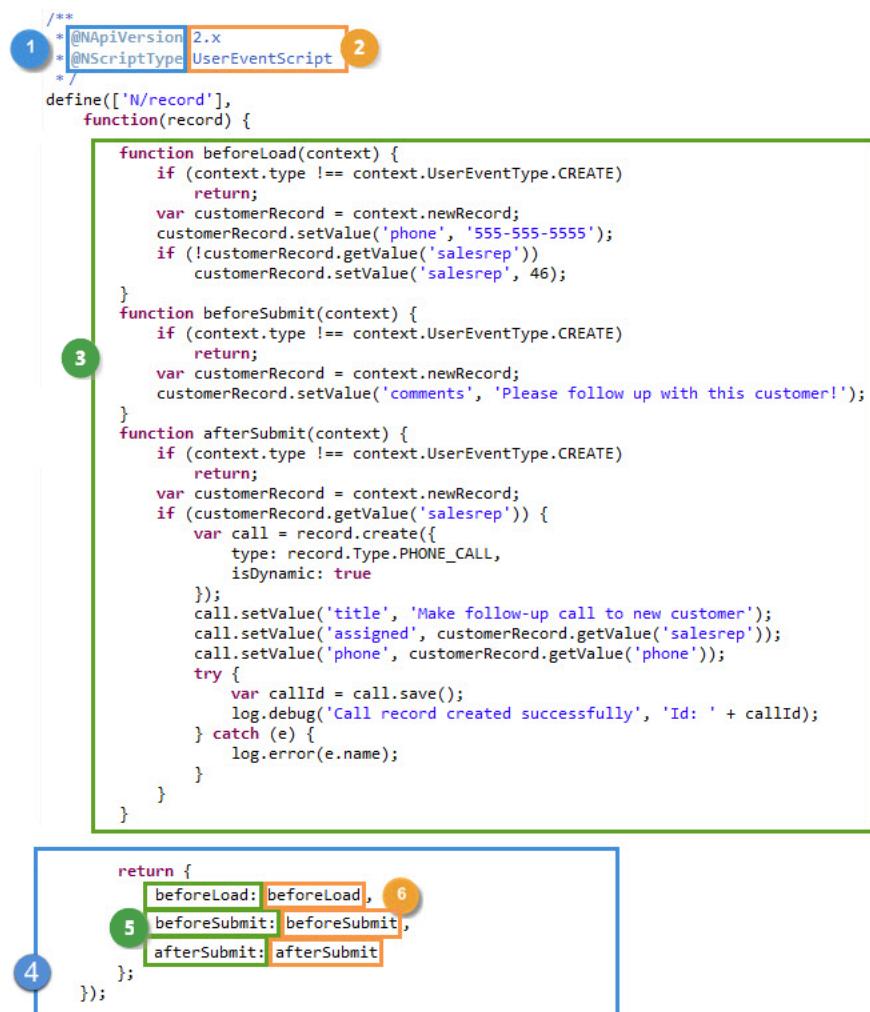
For details, see the following sections:

- [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.



Callout	Description
1	JSDoc tags
2	JSDoc tag values

Callout	Description
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 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

JSDoc Tag	Possible Values	Purpose	Possible Errors
			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 once 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.
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 once 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' view mode. 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_userevents), 'API VERSION' (2.0), and 'DESCRIPTION' (OWNER: John Smith, INACTIVE checked). A 'SCRIPT FILE' section shows 'preview newUserEventScript.js' with links for 'download' and 'Edit'. Below this are three checkboxes: 'BEFORE LOAD FUNCTION' (checked), 'BEFORE SUBMIT FUNCTION' (checked), and 'AFTER SUBMIT FUNCTION' (checked). The bottom navigation bar has tabs for 'Scripts' (which is selected), 'Parameters', 'Unhandled Errors', 'Execution Log', 'Deployments', and 'History'.

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.

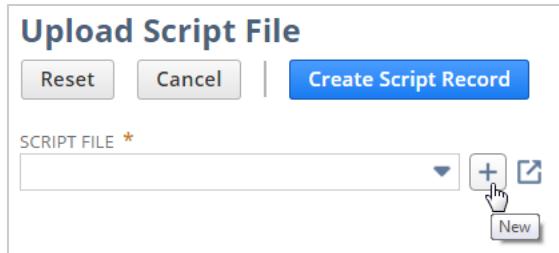
To create a script record:

1. Go to Customization > Scripting > Scripts > New.

The system displays the **Upload Script File** page.

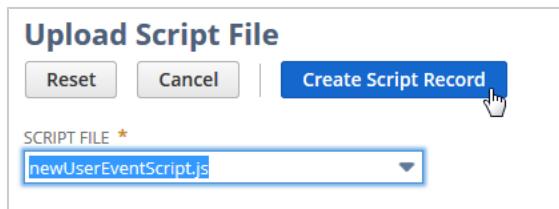
2. In the **Script File** dropdown list, select the appropriate SuiteScript 2.0 file. The dropdown list populates automatically with SuiteScript files that you have uploaded to your File Cabinet. This list includes version 1.0 and version 2.0 files.

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.

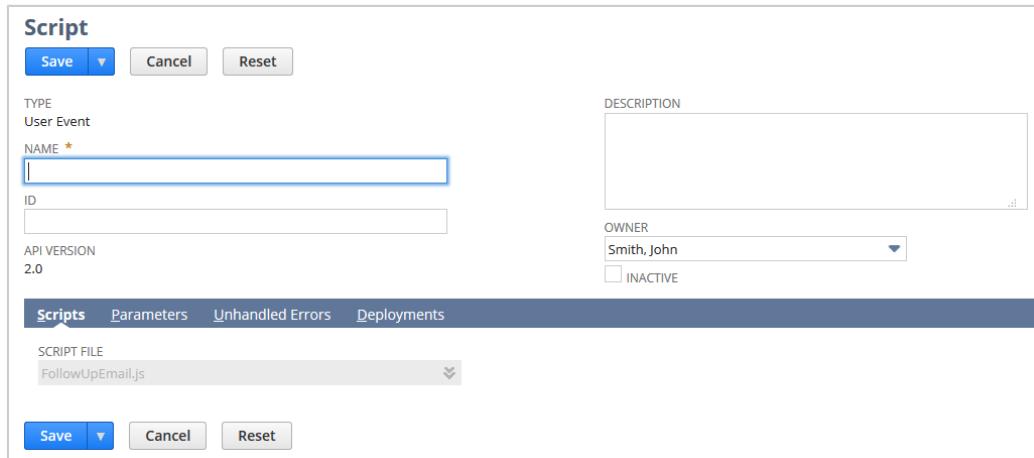


Click the icon to display a dialog box for uploading a file from your local environment.

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**.



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. 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 [Adding a Script Deployment 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 record at the time you create a script record, or you can deploy it later. The exact choices you make when deploying a script vary depending on its type. For example, with a user event script, you identify the record type the script applies to. For a scheduled script, you use the deployment record to identify the script's schedule.

Deployments are represented as lines on the Deployments subtab of the script record. For any deployment, you can also view a page that represents the deployment at Customization > Scripting > Script Deployments. Compared with the Deployments subtab, the Script Deployment page includes additional fields.

Multiple deployments can be created for the same script record. Deployments 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.

Note: The script deployment process is the same for a version 2.0 script as it is for a version 1.0 script.

For more details, see the following topics:

- [Methods of Creating a Script Deployment](#)
- [Adding a Script Deployment by Using the Deployments Sublist](#)
- [Editing a Script Deployment](#)

Methods of Creating a Script Deployment

To run a 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:

- Add a line to the script record's Deployments sublist, as described in [Adding a Script Deployment by Using the Deployments Sublist](#). This method lets you define values for most deployment fields, although not all. You can use this method at the following times:
 - At the time you are creating the script record.
 - For certain script types, as you are editing the script record. However, for some script types, the Deployments sublist is read only when you are editing the script record.
- Use the Script Deployment page, which displays all script deployment fields. You can open this page in the following ways:
 - Click the New Deployment button, which is available when you view the list of a script record's existing deployments. To display this list, go to Customization > Scripting > Scripts, and click the Deployments link for the appropriate script record. The New Deployment button is displayed at the top of the resulting page.

- When viewing an existing script deployment record, select Actions > New.
- Use nlapiCreateRecord with a type of scriptdeployment. For details, see the help topic [Script Deployment](#).

For details about the overall process of creating a script, uploading it, and deploying it, see [SuiteScript 2.0 Entry Point Script Deployment](#).

Adding a Script Deployment by Using the Deployments Sublist

If you are in the process of creating a script record, you can create a deployment by adding a row to the script record's Deployments sublist. For certain script types, you can also edit the script record and create a new deployment by adding a line to the Deployments sublist.

Be aware that the script deployment process varies somewhat depending on the script type. The following procedure describes basic steps common to all script types. For certain script types, additional steps may be required.

Note: There are additional methods of creating script deployments. For details, see [Script Deployment](#).

To create a script deployment by adding a line to the Deployments sublist:

1. 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.
2. Click the **Deployments** subtab.
3. 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**.
 - If appropriate, enter a value in the **ID** column. If you do not specify an ID, a system-generated ID is created for you.
 - In the **Deployed** column, leave the value set to its default, or click in the column to change it. For most script types, the default value is **Yes**. However, for map/reduce and scheduled scripts, the default is **Not Scheduled**.
 - 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. This link is displayed either in the **Title** or **Applies to** column. You can click the link to view the Script Deployment page for the deployment you just created. In some cases, this page lets you configure additional fields. For details, see [Editing a Script Deployment](#).

Editing a Script Deployment

If appropriate, you can edit a script deployment. For some script types, you can make changes by editing the script record's Deployments sublist. As an alternative, you can edit the script deployment by using the Script Deployment page. This procedure describes the latter method.

When you use the Script Deployment page, you have access to a greater number of fields than are available on the Deployments sublist. For example, the Script Deployment page includes the following subtabs:

- **Audience** – lets you specify particular roles that make up the script audience. When the script is deployed, it runs only in the accounts of the specified audience. In SuiteScript 2.0, this subtab is available for the following script types: client, portlet, RESTlet, Suitelet, and user event.
- **Links** – 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 edit 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**.

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.

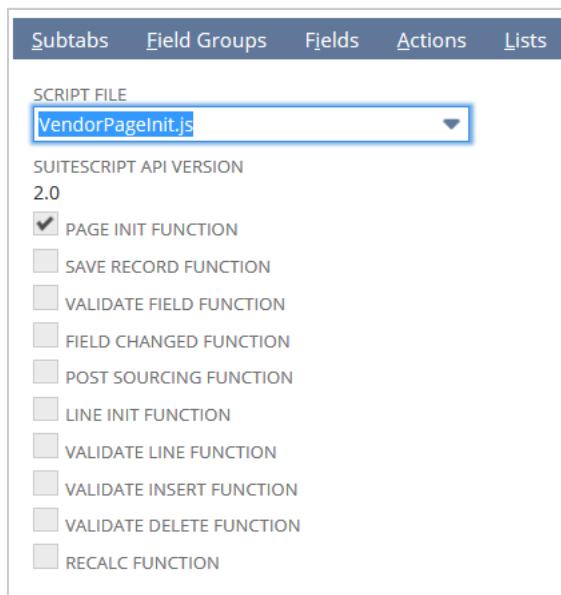
Important: Users must have at least the Edit level of the SuiteScript permission to attach a script to a custom form by editing the Custom Code tab of the form record. Users with the View level of the SuiteScript permission can see the Custom Code tab, but they cannot edit it.

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.

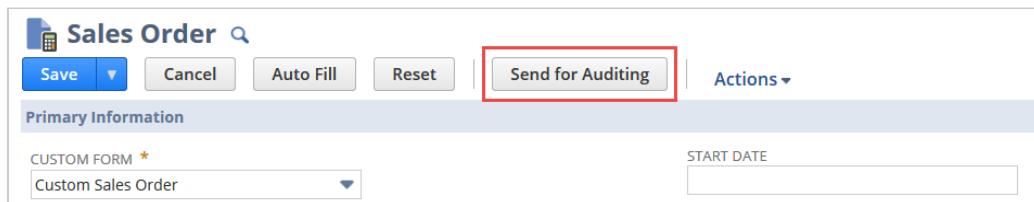
Note: If you edit the **Script File** field later and select a version 1.0 file instead of a version 2.0 file, the page reloads and includes fields into which you must manually enter the names of your entry-point functions. For details on this process, see the help topic [Step 3: Attach Script to Form](#).

4. Click **Save**.

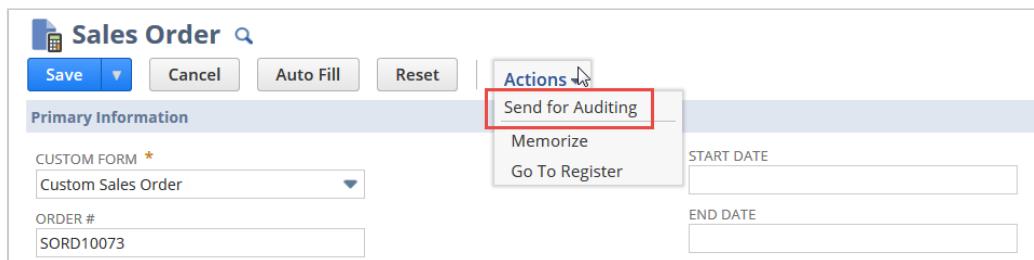
Configuring a Custom Action

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](#).