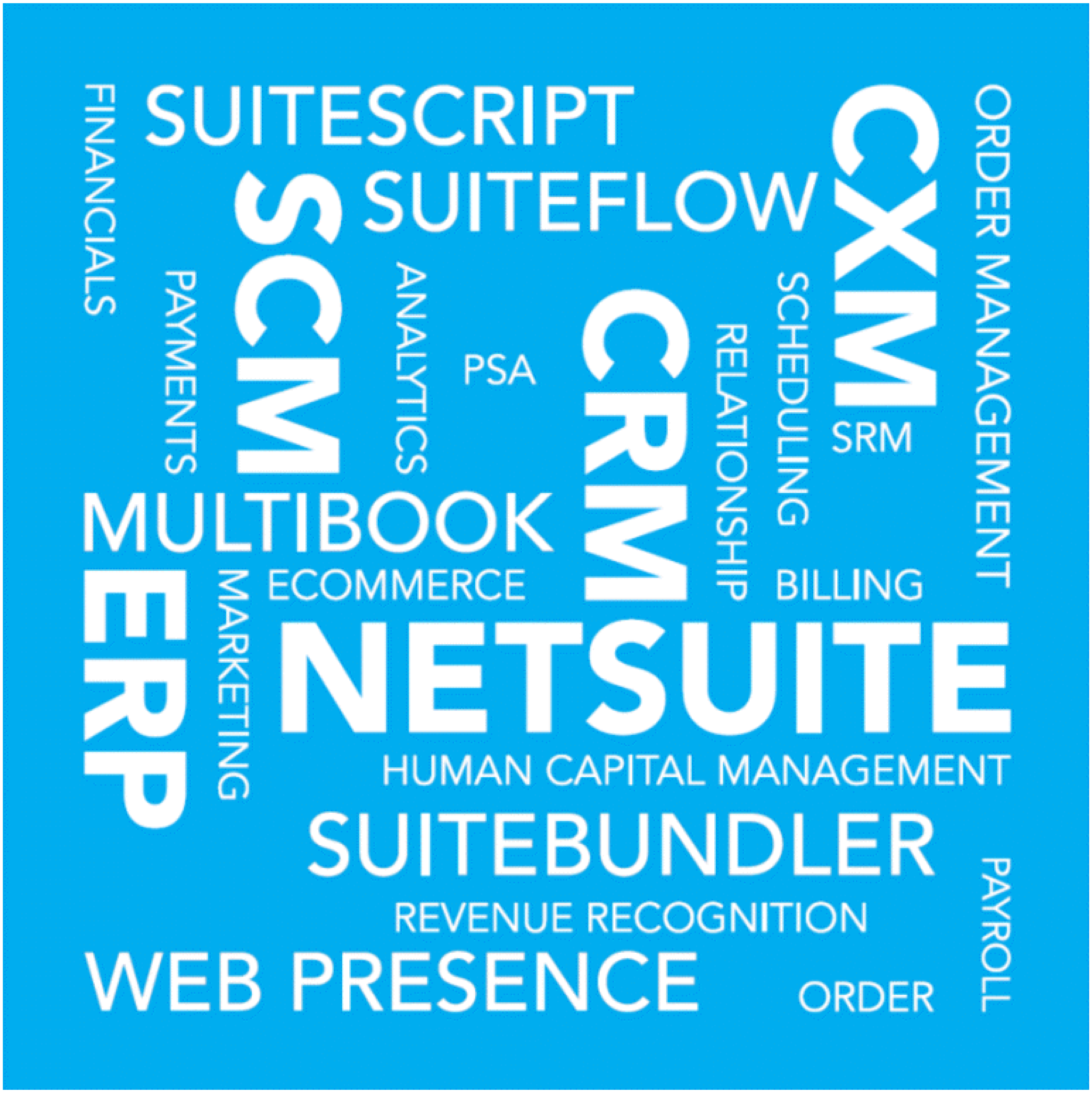
SuiteTalk (Web Services) Platform Guide



December 13, 2017 Version 2017 Release 2

Table of Contents

SuiteTalk Platform Overview

The SuiteTalk Platform provides programmatic access to your NetSuite data and business processes through an XML-based application programming interface (API). Generally speaking, the SuiteTalk Platform has the following characteristics:

- **SOAP encoded web services** – the SuiteTalk Platform uses SOAP-based web services with document style, or Doc style, encoding.

  Doc style encoding consists of message-oriented exchanges where an XML schema specified within the SOAP message defines the structure of any messages sent between two applications. RPC exchanges are NOT supported.
- **HTTPS Transport** – currently the only transport supported is HTTPS as defined in the WSDL document.

For a list of NetSuite records that are supported in web services development, see the help topic Web Services Supported Records. To see code samples of all SuiteTalk operations, see Web Services Operations.

To maintain the highest security standards for connectivity, NetSuite periodically updates server certificates and the trusted certificate store we use for https requests. We expect client applications that integrate with NetSuite to use an up-to-date certificate store, to ensure that integrations do not break because of certificates expiring or changing security standards. This issue typically does not affect modern browsers because these clients are maintained to use the latest certificates.

Important: Although the following material pertains to the 2017.2 WSDL, NetSuite continues to support endpoints released prior to 2017.2. For documentation about older endpoints that are supported, see the help topic SuiteTalk Archives.

In This Guide

This manual contains the following sections:

SuiteTalk Setup — provides steps for getting your web services development environment set up to create your first service. It also describes how to modify or set web services preferences. This section also describes general concepts that pertains to SuiteTalk development.

Web Services Preferences — provides information on setting company-wide preferences, request level preferences, search preferences, and setting the Internal ID preference.

Integration Management — provides information about managing external applications by using integration records.

Roles and Permissions in Web Services — defines the concept of a role within NetSuite and the necessity of providing a role ID during authentication. This section also describes how to assign a default role to a web services user, as well as how to set a “Web Services Only” role for a user.

Records, Fields, Forms, and Sublists in Web Services — describes how to work with records, field, and sublist objects in web services. This section also provides information on working with custom forms.

Web Services Processing — describes how to process requests synchronously versus asynchronously. Also provided are steps for monitoring your web services requests.

Web Services Security — describes all aspects of web services security and session management.

Platform Features — describes the Web Services Concurrent License.

Types — describes the various types available in the SuiteTalk Platform.

Web Services Operations — describes each operation that can be performed through web services and provides SOAP, C# and Java code samples.

Web Services PHP Toolkit — describes the tools that assist in the development of PHP applications that interface to the SuiteTalk Platform.

Web Services Error Handling and Error Codes — provides tables of possible SOAP faults, fault status codes and error status codes that can be thrown for a particular request.

Support Has Ended for SuiteTalk 2009.2 and Earlier Endpoints

NetSuite no longer supports SuiteTalk (Web Services) 2009.2 or earlier endpoints. NetSuite strongly encourages customers and partners using 2009.2 and earlier versions of the SuiteTalk WSDL to begin upgrading to later endpoints.

The end of support for these endpoints is enabling NetSuite to focus maintenance on fewer versions, and will enable customers to benefit from more consistent record processing behavior in the current endpoints.

Determining Endpoints that are Currently in Use

Use the following guidelines to determine whether any of your SuiteTalk integrations use the 2009.2 or earlier endpoints:

- For each partner application, work with the partner directly to find out the endpoint in use.
- For your own integrations:
  - You can check your SOAP requests to see which endpoints they reference. You can find your requests in two places: on the Execution Log subtab of the appropriate integration record, or at Setup > Integration > Web Services Usage Log. The Web Services Usage Log shows all requests, regardless of which application sent them. By contrast, each integration record would show the requests sent only by the application linked to that specific integration record.
  - You can contact NetSuite Support to ask for help in determining the endpoints used in your account.

Note that different integrations in your account may use different endpoints.

Resources for Endpoint Upgrades

To begin the process of upgrading your endpoint, refer to the following resources.

- For the schema of the latest endpoint, see:

  https://webservices.netsuite.com/wsdl/v2017_2_0/netsuite.wsdl
- For details about the records supported in the latest endpoint, see the help topic SuiteTalk Records Overview.
- For general information about using SuiteTalk, see SuiteTalk Platform Overview.
- For upgrade tips, see Upgrading WSDL Versions.

- If you have issues during your transition process, please contact NetSuite Technical Support.

Note About Record Processing Behavior in 2010.1 and Later Endpoints

Be aware of the following change: When modifying records using the 2009.2 or an earlier endpoint, most, but not all, of the NetSuite core business logic is executed. For integrations that use an endpoint later than 2009.2, all core business logic executes when the record is submitted. So you may notice some changes in SuiteTalk behavior for integrations that are upgraded to use an endpoint later than 2009.2. Following your upgrade, the behavior will more closely match the behavior when records are modified through the NetSuite user interface.

Web Services Overview

Web services are Extensible Markup Language (XML) applications mapped to programs, objects, databases or complex business functions. They utilize a standardized XML messaging system to send or receive requests to authorized parties over the Internet. Businesses can implement web services to provide standards based processes that can be utilized by other organizations or integrated with business partner processes. Since the programming logic encapsulated by each Web service is independent of any one platform or technology set, web services provide an independence and flexibility for integration across and between businesses.

The following protocols are used to publish, expose or access web services:

- WSDL (Web Services Description Language): a WSDL file exposes a Web service interface, interaction patterns and protocol mapping. WSDL can be readily interpreted by other applications, systems and platforms.
- UDDI (Universal Description, Discovery and Integration): A Web Service can be categorized and registered in a UDDI Registry so that applications can locate it and retrieve its WSDL. Currently NetSuite does NOT provide a UDDI registry.
- SOAP (Simple Object Access Protocol): A Web Service can use the SOAP messaging format to define an envelope for web services communication via HTTP, HTTPs or other transport layers.

With the SuiteTalk Platform, you can integrate existing business functions within your organization to NetSuite through the use of web services. Web services can be invoked in real-time to perform operations such as retrieving, adding, updating, and deleting data.

NetSuite WSDL and XSD Structure

The SuiteTalk Platform has a single WSDL file that describes all supported operations and messages. You can access that file at the following link:

https://webservices.netsuite.com/wsdl/v2017_2_0/netsuite.wsdl

In this link, v2017_2_0 reflects the WSDL version.

NetSuite defines WSDL versioning, the location of schemas, namespaces, and the endpoint as follows:

```
WSDL: https://webservices.netsuite.com/wsdl/v2017_1_0/netsuite.wsdl
```

```
   <xsd:import namespace="urn:core_2017_1_0.platform.webservices.netsuite.com"
  schemaLocation="https://webservices.netsuite.com/xsd/platform/v2017_1_0/core.xsd"/>
<port name="NetSuitePort" binding="tns:NetSuiteBinding">
   <soap:address location="https://webservices.netsuite.com/services/NetSuitePort_2017_1_0" />

</port>
```

Note: For more information on WSDL versioning, upgrading, and testing, see NetSuite WSDL Versioning. For information on WSDLs that pre-date 2017.2, see the help topic SuiteTalk Archives.

The WSDL is composed of numerous NetSuite-specific types that are defined in related XSDs. Each XSD URL has an alias that can be used as a reference to the corresponding XSD file. The following tables show the organization of the XSD files.

Important: NetSuite hosts customer accounts in multiple data centers. For that reason, the correct URL for web services access varies depending on the data center hosting the account. Your integration **must** incorporate logic that dynamically determines the correct URL. With the 2012.2 and later endpoints, you should use the getDataCenterUrls operation to dynamically discover the correct URL. With older endpoints, you should use the REST roles service. For details, see the help topic Using the REST roles Service to Get User Accounts, Roles, and Domains. You can also use the DataCenterUrls REST Service to dynamically discover the correct domain URLs.

Messaging XSD Files

These files provide descriptions for the base web services functions used by all operations. For API documentation on each operation, see Web Services Operations.

URL	Schema Alias	Notes
https://webservices.netsuite.com/xsd/platform/v2017_2_0/common.xsd	platformCommon	This file also includes descriptions for the following subrecords: ■ Address ■ InventoryDetail ■ LandedCost
https://webservices.netsuite.com/xsd/platform/v2017_2_0/core.xsd	platformCore	Among other elements, this file includes descriptions for the following complex types, which are used for authentication: ■ Passport ■ SsoPassport ■ TokenPassport
https://webservices.netsuite.com/xsd/platform/v2017_2_0/faults.xsd	platformFaults	
https://webservices.netsuite.com/xsd/platform/v2017_2_0/messages.xsd	platformMsgs	

Record Type Definitions

The following XSD files provide descriptions for each record type in NetSuite. For field reference information on each record, see the help topic SuiteTalk Records Overview. For general information on working with records in SuiteTalk, see Records in Web Services.

URL	Schema Alias	Record Types
https://webservices.netsuite.com/xsd/activities/v2017_2_0/scheduling.xsd	actSched	ContactCategory CustomerCategory SalesRole PriceLevel WinLossReason Term NoteType PaymentMethod CalendarEvent CalendarEventSearch CalendarEventSearchAdvanced Task TaskSearch TaskSearchAdvanced PhoneCall PhoneCallSearch PhoneCallSearchAdvanced ProjectTask Resource Allocation
https://webservices.netsuite.com/xsd/transactions/v2017_2_0/demandPlanning.xsd	demandPlanning	ItemDemandPlan ItemSupplyPlan
https://webservices.netsuite.com/xsd/documents/v2017_2_0/fileCabinet.xsd	docfileCab	File FileSearch FileSearchAdvanced Folder FolderSearch FolderSearchAdvanced
https://webservices.netsuite.com/xsd/general/v2017_2_0/communication.xsd	generalComm	Note NoteSearch NoteSearchAdvanced Message MessageSearch MessageSearchAdvanced
https://webservices.netsuite.com/xsd/lists/v2017_2_0/accounting.xsd	listAcct	Billing Schedule ContactCategory CustomerCategory SalesRole PriceLevel WinLossReason Term NoteType PaymentMethod LeadSource Assembly Item (BOM Item) DescriptionItem DiscountItem

URL	Schema Alias	Record Types
		Download Item GiftCertificateItem InventoryItem ItemSearch ItemSearchAdvanced Kit/Package Item Lot Numbered Assembly Item Lot Numbered Inventory Item MarkupItem NonInventoryPurchaseItem NonInventoryResaleItem NonInventorySaleItem OtherChargePurchaseItem OtherChargeResaleItem OtherChargeSaleItem PaymentItem SerializedAssemblyItem Serialized Inventory Item ServicePurchaseItem ServiceResaleItem ServiceSaleItem SubtotalItem Currency ExpenseCategory Account AccountSearch AccountSearchAdvanced AccountingPeriod Bin BinSearch BinSearchAdvanced Classification ClassificationSearch ClassificationSearchAdvanced Department DepartmentSearch DepartmentSearchAdvanced Expense Category Gift Certificate FairValuePrice
https://webservices.netsuite.com/xsd/lists/v2017_2_0/accounting.xsd	listAcct	GiftCertificateSearch GiftCertificateSearchAdvanced GlobalAccountMapping Inventory Number ItemAccountMapping Location LocationSearch LocationSearchAdvanced Nexus Revenue Recognition Schedule Revenue Recognition Template Sales Tax ItemItemSearchAdvanced ContactRole Bin SalesTaxItem

URL	Schema Alias	Record Types
		TaxAcct TaxGroup TaxType Subsidiary SubsidiarySearch SubsidiarySearchAdvanced UnitsType PartnerCategory VendorCategory
https://webservices.netsuite.com/xsd/lists/v2017_2_0/employees.xsd	listEmp	Employee EmployeeSearch EmployeeSearchAdvanced HCM Job
https://webservices.netsuite.com/xsd/lists/v2017_2_0/marketing.xsd	listMkt	Campaign CampaignSearch CampaignSearchAdvanced CampaignCategory CampaignAudience CampaignFamily CampaignSearchEngine CampaignChannel CampaignOffer CampaignResponse CampaignVertical CampaignSubscription PromotionCode PromotionCodeSearch PromotionCodeSearchAdvanced
https://webservices.netsuite.com/xsd/lists/v2017_2_0/relationships.xsd	listRel	Billing Account Contact ContactSearch ContactSearchAdvanced Customer CustomerSearch CustomerSearchAdvanced CustomerStatus Partner PartnerSearch PartnerSearchAdvanced Vendor VendorSearch VendorSearchAdvanced EntityGroup EntityGroupSearch EntityGroupSearchAdvanced Job JobSearch JobSearchAdvanced JobType JobStatus
https://webservices.netsuite.com/xsd/lists/v2017_2_0/supplyChain.xsd	listScm	ManufacturingCostTemplate ManufacturingOperationTask ManufacturingRouting

URL	Schema Alias	Record Types
https://webservices.netsuite.com/xsd/lists/v2017_2_0/website.xsd	listSite	SiteCategory SiteCategorySearch SiteCategorySearchAdvanced
https://webservices.netsuite.com/xsd/lists/v2017_2_0/support.xsd	listSupport	SupportCase SupportCaseSearch SupportCaseSearchAdvanced SupportCaseStatus SupportCaseType SupportCaseOrigin SupportCaseIssue SupportCasePriority Solution SolutionSearch SolutionSearchAdvanced Topic TopicSearch TopicSearchAdvanced Issue IssueSearch IssueSearchAdvanced
https://webservices.netsuite.com/xsd/setup/v2017_2_0/customization.xsd	setupCustom	CustomRecord CustomRecordSearch CustomRecordSearchAdvanced CustomList CustomRecordType EntityCustomField CrmCustomField OtherCustomField ItemCustomField TransactionBodyCustomField TransactionColumnCustomField ItemOptionCustomField CustomRecordCustomField CustomTransaction
https://webservices.netsuite.com/xsd/transactions/v2017_2_0/bank.xsd	tranBank	Check Deposit
https://webservices.netsuite.com/xsd/transactions/v2017_2_0/customers.xsd	tranCust	CashRefund Charge CustomerPayment ReturnAuthorization CreditMemo CustomerRefund CustomerDeposit DepositApplication
https://webservices.netsuite.com/xsd/transactions/v2017_2_0/inventory.xsd	tranInvt	InventoryAdjustment InventoryCostRevaluation AssemblyBuild AssemblyUnbuild WorkOrder WorkOrderClose WorkOrderIssue WorkOrderCompletionOperation

URL	Schema Alias	Record Types
https://webservices.netsuite.com/xsd/transactions/v2017_2_0/employees.xsd	tranEmp	TimeBill TimeBillSearch TimeBillSearchAdvanced ExpenseReport TimeEntry Timesheet Paycheck
https://webservices.netsuite.com/xsd/transactions/v2017_2_0/financial.xsd	tranFin	Budget BudgetSearch BudgetSearchAdvanced
https://webservices.netsuite.com/xsd/transactions/v2017_2_0/general.xsd	tranGeneral	JournalEntry InterCompanyJournalEntry StatisticalJournalEntry AdvIntercompanyJournalEntry
https://webservices.netsuite.com/xsd/transactions/v2017_2_0/purchases.xsd	tranPurch	VendorBill PurchaseOrder ItemReceipt VendorPayment VendorCredit Purchase/Requisition InboundShipment
https://webservices.netsuite.com/xsd/transactions/v2017_2_0/sales.xsd	tranSales	AccountingTransactionSearch AccountingTransactionSearchAdvanced Opportunity OpportunitySearch OpportunitySearchAdvanced SalesOrder TransactionSearch TransactionSearchAdvanced ItemFulfillment Invoice CashSale Estimate

System Constants XSD Files

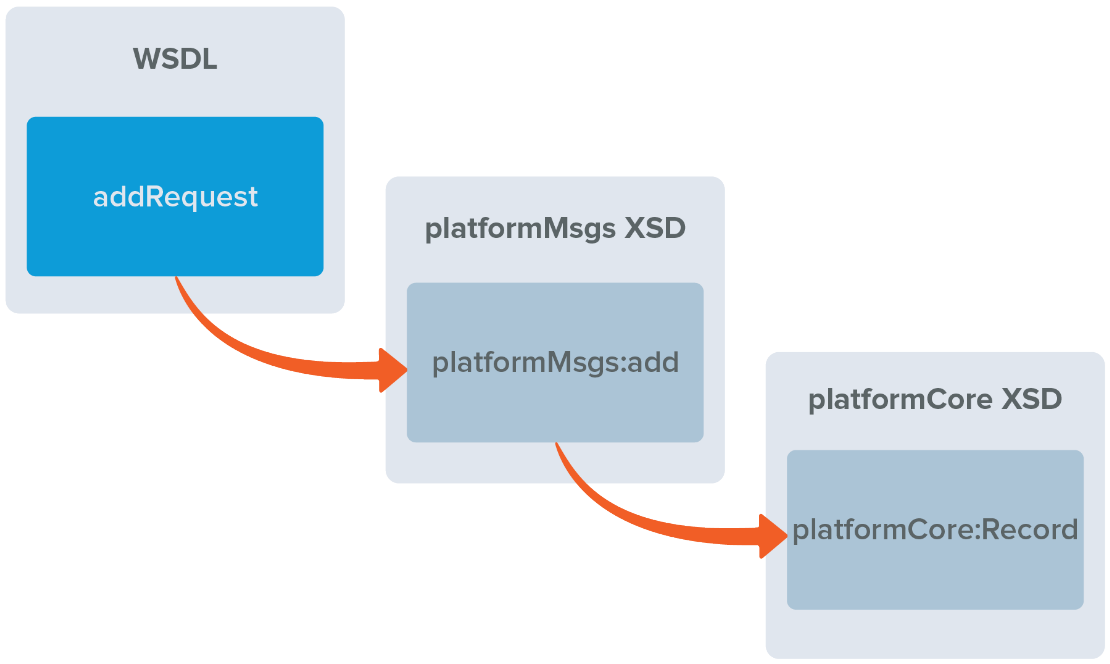
These files provide constant values for the corresponding types in the business records XSD files.

URL	Schema Alias
https://webservices.netsuite.com/xsd/activities/v2017_2_0/schedulingTypes.xsd	actSchedTyp
https://webservices.netsuite.com/xsd/transactions/v2017_2_0/demandPlanning.xsd	demandPlanningTyp
https://webservices.netsuite.com/xsd/documents/v2017_2_0/fileCabinetTypes.xsd	docFileCabTyp
https://webservices.netsuite.com/xsd/general/v2017_2_0/communicationTypes.xsd	generalCommTyp

URL	Schema Alias
https://webservices.netsuite.com/xsd/transactions/v2017_2_0/inventoryTypes.xsd	invtTyp
https://webservices.netsuite.com/xsd/lists/v2017_2_0/accountingTypes.xsd	listAcctTyp
https://webservices.netsuite.com/xsd/lists/v2017_2_0/employeeTypes.xsd	listEmpTyp
https://webservices.netsuite.com/xsd/lists/v2017_2_0/marketingTypes.xsd	listMktTyp
https://webservices.netsuite.com/xsd/lists/v2017_2_0/relationshipTypes.xsd	listRelTyp
https://webservices.netsuite.com/xsd/lists/v2017_2_0/supplyChainTypes.xsd	listScmTyp
https://webservices.netsuite.com/xsd/lists/v2017_2_0/supportTypes.xsd	listSupportTyp
https://webservices.netsuite.com/xsd/platform/v2017_2_0/coreTypes.xsd	platformCoreTyp
https://webservices.netsuite.com/xsd/platform/v2017_2_0/faultTypes.xsd	platformFaultsTyp
https://webservices.netsuite.com/xsd/setup/v2017_2_0/customizationTypes.xsd	setupCustomTyp
https://webservices.netsuite.com/xsd/transactions/v2017_2_0/saleTypes.xsd	tranSalesTyp

Example

For example, the addRequest message type has three levels of referencing.



In the WSDL file, the **addRequest** message is defined as:

```
<message name="addRequest">
   <part name="parameters" element="platformMsgs:add"/>
</message>
```

The element called **platformMsgs:add** is defined in the platformMsgs XSD file. In this case, the platformMsgs alias refers to the xsd file at:

https://webservices.netsuite.com/xsd/platform/v2017_2_0/messages.xsd

In this file, the addRequest element is defined again as:

```
<complexType name="AddRequest">
   <sequence>
       <element name="record" type="platformCore:Record" />
    </sequence>
</complexType>
```

Again there is a reference that is not contained in this XSD file called **platformCore:Record**. The platformCore alias refers to the XSD file at:

https://webservices.netsuite.com/xsd/platform/v2017_2_0/core.xsd

The abstract type Record is defined as:

```
<complexType name="Record" abstract="true">
   <sequence>
      <element name="nullFieldList" type="platformCore:NullField" minOccurs="0" maxOccurs="1" /
>
   </sequence>
</complexType>
```

Note: In the SuiteTalk Platform, Record is the base for all other NetSuite record types.

NetSuite WSDL Versioning

The following topics provide information on NetSuite WSDL versioning, upgrading to a new WSDL version, and things to consider when your NetSuite account is upgraded. If you are new to SuiteTalk development, it is recommended that you read these topics in order:

- NetSuite Versioning and WSDL Versioning Overview
- Upgrading WSDL Versions
- Testing After a NetSuite Upgrade
- Support for Existing WSDL Versions

NetSuite Versioning and WSDL Versioning Overview

The naming convention for NetSuite versions and WSDL versions are the same. For example, for Version 2017 Release 2, there is an accompanying 2017_2 version of the WSDL, as shown in the following WSDL URL:

https://webservices.netsuite.com/wsdl/v2017_2_0/netsuite.wsdl

NetSuite WSDL versioning denotes:

- the major version: **v2017_2**
- the minor or patch version: **_0**

Generally speaking, the WSDL version that accompanies a NetSuite release incorporates much of the new functionality offered in the NetSuite upgrade. For example, if new workflows or new records are introduced in a NetSuite upgrade, the accompanying WSDL often includes new operations or records to support the NetSuite enhancements. In some releases, a new WSDL also includes enhancements to the SuiteTalk platform itself. These enhancements are independent of NetSuite product enhancements.

Be aware that when your NetSuite account is upgraded, **you do not have to upgrade** your WSDL to the supporting NetSuite version. For example, you may still use version 2017.1 of the WSDL in conjunction with a NetSuite account that has been upgraded to Version 2017 Release 2. For more information, see Upgrading WSDL Versions.

With each new release of NetSuite (and new WSDL version), it is recommended that you read the SuiteTalk portion of the Sneak Peeks and the Release Notes that come with the release. These documents list all of the new functionality and any schema bug fixes offered in the latest WSDL.

Upgrading WSDL Versions

NetSuite customers typically upgrade their WSDL when a newer version includes functionality that enables them to meet a particular business need. They will also upgrade their WSDL when the one they are currently using is going to be retired by NetSuite (see Support for Existing WSDL Versions for details). If you do plan to upgrade the WSDL you are using, NetSuite strongly recommends that you upgrade to the latest available version to prolong the life cycle for your integrated application.

Important: You are not required to upgrade your WSDL when your NetSuite account is upgraded. However, when your account is upgraded, you should re-test all existing integrations to ensure they work against the latest version of NetSuite (see Testing After a NetSuite Upgrade for details).

After upgrading your WSDL version, any API incompatibilities between the new and old versions will be displayed in your development environment. You will be unable to proceed to unit testing until you have fixed any IDE compilation errors. Consequently, most WSDL upgrade issues are usually addressed during development.

Upgrading your WSDL version may require you to re-code parts of your integrated application to be compatible with the latest version of the NetSuite product. Re-coding may be due to any of the following reasons:

- a new required field has been added to NetSuite
- a field that was optional in previous versions of NetSuite becomes required in the latest version
- a field that is referenced in your integration has been removed from NetSuite
- the data type for a field has changed
- the method signature changes for a SuiteTalk operation that is used in your integration

Note: See Working with the Araxis Merge Diff File for information on how to use Araxis to see the changes made from one WSDL version to the next. You can also get an idea of the types of changes you must make by reading the release notes from the endpoint you are currently using onwards to the last release of the WSDL.

Testing After a WSDL Upgrade

If you are upgrading your WSDL to make use of a particular NetSuite enhancement, it is recommended that you use the UI to learn how the feature works. In most cases, NetSuite web services is the programmatic equivalent of the NetSuite UI. Therefore, if you know how a feature works in the UI, your learning curve for understanding how to leverage the feature in web services will be minimized.

Be aware that WSDL upgrades may take considerable planning and testing. Consequently, you may decide to upgrade only when you know your version is going to be retired. (See Support for Existing WSDL Versions for more information.)

Testing After a NetSuite Upgrade

NetSuite strives to maintain backward compatibility from one NetSuite version to the next. However, your integration may rely on unique workflows that might break when your NetSuite account is upgraded. Therefore, NetSuite recommends that you test all existing integrations in your Release Preview account for the NetSuite upgrade and report any issues you may find to ensure a smooth upgrade of your production account.

In your Release Preview (or beta) account, you also must modify the WSDL references you are using in your application to point to one of the following endpoints (based on the endpoint you are currently using):

- https://webservices.beta.netsuite.com/wsdl/v2017_2_0/netsuite.wsdl
- https://webservices.beta.netsuite.com/wsdl/v2017_1_0/netsuite.wsdl
- https://webservices.beta.netsuite.com/wsdl/v2016_2_0/netsuite.wsdl
- https://webservices.beta.netsuite.com/wsdl/v2016_1_0/netsuite.wsdl

- https://webservices.beta.netsuite.com/wsdl/v2015_2_0/netsuite.wsdl
- https://webservices.beta.netsuite.com/wsdl/v2015_1_0/netsuite.wsdl
- https://webservices.beta.netsuite.com/wsdl/v2014_2_0/netsuite.wsdl
- https://webservices.beta.netsuite.com/wsdl/v2014_1_0/netsuite.wsdl
- https://webservices.beta.netsuite.com/wsdl/v2013_2_0/netsuite.wsdl
- https://webservices.beta.netsuite.com/wsdl/v2013_1_0/netsuite.wsdl
- https://webservices.beta.netsuite.com/wsdl/v2012_2_0/netsuite.wsdl
- https://webservices.beta.netsuite.com/wsdl/v2012_1_0/netsuite.wsdl
- https://webservices.beta.netsuite.com/wsdl/v2011_2_0/netsuite.wsdl
- https://webservices.beta.netsuite.com/wsdl/v2011_1_0/netsuite.wsdl
- https://webservices.beta.netsuite.com/wsdl/v2010_2_0/netsuite.wsdl
- https://webservices.beta.netsuite.com/wsdl/v2010_1_0/netsuite.wsdl

Note: If you are engaged with a partner for any of your web services integrations, it is recommended that you contact them and inform them of when your Release Preview (beta) account will be available. They can then set up a time with you and a test plan for the integration.

Working with the Araxis Merge Diff File

To help you see the differences between the latest WSDL and the version that preceded it, refer to the Araxis diff file provided in the release notes for each release. When reviewing the diff file, start by investigating the records you are currently using. For example, if your web services application interacts with customer and vendor records, use the diff file to review any changes made to these records. These changes may include added or removed fields or field (element) data type changes.

If you want to compare the latest WSDL with the version you are currently using, you can download Araxis and compare your own version to any other version listed on the Developer Portal.

For more information on using the Araxis diff file to review NetSuite WSDL changes, see the help topic Araxis Merge Diff File for the 2017.2 Endpoint, in the Version 2017 Release 2 Release Notes.

Support for Existing WSDL Versions

NetSuite WSDLs are supported for up to three years from the date of first release. After this time, NetSuite recommends that users upgrade to the latest WSDL version. To find the latest version, see the help topic SuiteTalk Archives.

Note that if you do not upgrade when your WSDL is retired, your integrations may cease to run as intended and the endpoint for that version may be shut down.

Web Services Governance Overview

To optimize NetSuite application and database servers, the system employs certain mechanisms to control the consumption of web services. These mechanisms ensure the following:

- Requests are monitored and controlled to ensure that the user experience is not excessively impacted.

- The burden of heavy web services users is not shared among all users.

NetSuite web services governance includes:

- Record limiting (see Understanding Record Limiting)
- Request limiting (see Understanding Request Limiting)

Note: For information on the maximum number of sessions, see Session Limits.

Important: For information about the new web services and RESTlet concurrency governance, see Web Services and RESTlet Concurrency Governance Starting From Version 2017.2.

Understanding Record Limiting

Limits exist for the following:

- The number of records that can be included in a list operation.
- The number of records that can be included in a page of search results. Search results are set by the pageSize field of the searchPreferences complex type. A minimum and maximum exists for this parameter.

The limit varies depending on whether the operation is synchronous or asynchronous.

Synchronous Operations

Operation or parameter (on a per request basis)	Minimum Record Count	Maximum Record Count
addList		200
deleteList		200
getList		1000
getItemAvailability		10000
updateList		100
upsertList		100
pageSize parameter of searchPreferences complex type, when used in synchronous searches	5	1000

Asynchronous Operations

Operation or parameter (on a per request basis)	Minimum Record Count	Record Count
asyncAddList		400
asyncGetList		2000
asyncUpdateList		200

Operation or parameter (on a per request basis)	Minimum Record Count	Record Count
asyncUpsertList		200
asyncDeleteList		400
pageSize parameter of searchPreferences complex type, when used in asynchronous searches	5	2000

Note: For details on asynchronous processing, see Asynchronous Request Processing.

Understanding Request Limiting

Request limiting pertains to the size of your SOAP request. The size of a request cannot exceed 100MB.

Understanding Governance Errors

The following faults are thrown because of other governance violations.

- **ExceededRecordCountFault** – thrown if a request exceeds the allowed record count (see Understanding Record Limiting).
- **ExceededRequestLimitFault** – thrown if the allowed number of concurrent requests is exceeded (see Understanding Request Limiting).
- **ExceededRequestSizeFault** – thrown if a request exceeds 100M (see Understanding Request Limiting).

For more information on exceptions, refer to Web Services Error Handling and Error Codes.

Web Services and RESTlet Concurrency Governance Starting From Version 2017.2

In previous releases of NetSuite, concurrency for web services and RESTlets was governed separately per user and authentication method. Starting from version 2017.2, web services and RESTlet concurrency is also governed per account. The new account governance limit applies to the combined total of web services and RESTlet requests.

Note: For a comparison of the previous and the new governance model, see the NetSuite Governance Changes PDF file.

Timing of the Change

The timing of the change is different for different account types.

The change is enforced in all 2017.2 release preview accounts, and in all development account, as of the 2017.2 upgrade.

The change is enforced for most production and sandbox accounts as of the 2017.2 upgrade. For a subset of accounts that we have determined may be affected by this change, the change to concurrency governance is not tied to the 2017.2 upgrade. For these accounts, account administrators receive separate communications with details about the timing of the change.

Details of the Change

The governance limit for concurrent requests is based on the service tier and the number of SuiteCloud Plus licenses available for your production, sandbox, and release preview accounts. The base limit is increased by 10 for each SuiteCloud Plus license.

Important: For development and partner accounts, the base limit is always five, and the limit does not scale with the number of SuiteCloud Plus licenses.

Service Tier	Account Base Limit
Shared, 3	5
2	10
1, 1+, 0	15

For example, if you are on Service Tier 1 and you have five SuiteCloud Plus licenses, the limit for concurrent requests in your production account is 65 (15+5*10). If you are on shared service with one SuiteCloud Plus license, your account limit is 15 (5+10).

For information about service tiers and SuiteCloud Plus licenses, see this data sheet.

For web services that use the login and logout operations or request-level credentials for authentication, the existing governance limits per user are kept. For web services that use token-based authentication (TBA) and for RESTlets, there are no per-user limits for concurrent requests.

Note: For sessions authenticated with Outbound Single Sign-on (SuiteSignOn), concurrency is governed per user and additionally per account.

Note: For details about different authentication types, see Authentication for Web Services.

The change in governance can provide increased concurrency for web services integrations that use TBA. This change also means more limits on RESTlet integrations, which are capped at the maximum concurrency defined for an account.

If you exceed the limit for concurrent requests, an error is thrown.

You can find more information about your account concurrency limits and web services usage by using the following resources:

- The account concurrency limit for your account is displayed on the Web Services Preferences page, which you can access at Setup > Integration > Web Services Preferences (Administrator).
- The Web Services Preferences page also displays information about whether the account concurrency governance is enabled in your account.
- The Web Services Execution Log and Web Services Usage Logs contain information about the requests that were rejected due to a violation of the concurrency governance limits. For information about accessing the logs, see Web Services Execution Log and Using the Web Services Usage Log.
- The results of a web services operations search also contain information about web services operations that are rejected due to concurrency violation. For more information, see Web Services Operations Search Type.

- The Application Performance Management (APM) SuiteApp includes a web services performance dashboard. Using this dashboard, you can monitor web services record processing and the status of web services operations, and find out details about the operations that were rejected. For general information about APM, see the help topic Application Performance Management (APM). For information about monitoring web services operations using APM, see the help topic Analyzing Web Services Performance.

Note: For information about viewing rejected RESTlet requests, see the help topic Viewing a List of Script Execution Logs

RESTlet and web services requests that are rejected because an account has exceeded the concurrency governance limit result in one of the following server responses.

For RESTlet requests, the following errors occur:

- HTTP error code: 400 Bad Request
- SuiteScript error code: SSS_REQUEST_LIMIT_EXCEEDED

For web services requests, the following SOAP faults occur:

Authentication Type	SOAP Fault	Error Message
Request-level credentials Login and logout operations	ExceededRequestLimitFault	WS_CONCUR_SESSION_ DISALLWD
TBA	ExceededConcurrentRequestLim itFault	WS_REQUEST_BLOCKED

Sample Scenario

Consider Customer ABC who are on a shared service tier with one SuiteCloud Plus license. User A is the designated concurrent web services user. Customer ABC runs 10 SOAP calls concurrently. This uses request-level credentials with User A as the user. They also have one RESTlet that is hit concurrently five times. It uses token-based authentication (TBA). They also use User B to generate one web service request using request-level credentials. All of these occur in parallel.

Starting from version 2017.2, their account-level concurrency limit will be 15 (5 +10). If they generate five RESTlet requests using TBA, 10 SOAP requests with request-level credentials with User A, and one SOAP request with request-level credentials with User B, an error will be thrown for one of the requests.

This is because they generated one request in excess of their account-level limit. Therefore, 15 requests will succeed, one request will result in an error. While rare, this can occur for any of the requests based on a race condition. If they do not want to use a dedicated concurrent web services user, they can re-implement their SOAP integrations to use TBA instead of request-level credentials. They will still need the SuiteCloud Plus license to get 15 web services concurrency.

Recommended Actions

Review your client applications to ensure they can handle the error codes. The following sample is an example of how you might handle the error codes programmatically.

```
maxAttempts = 5 // try it 5 times, then fail for good
```

```
while (i < maxAttempts)
{
response = do_WS_call()
isSuccess = response.getIsSuccess()
errorMsg = response.getErrorMsg()

if !(isSuccess == FALSE && (errorMsg == WS_CONCUR_SESSION_DISALLWD || errorMsg == WS_REQUEST_BL
OCKED))
then
    break // end the cycle
end if

wait()
i++
}
```

Additionally, to ensure that the requests of non-concurrent users do not overlap, upgrade your client applications to serialize your requests.

If your client application receives the WS_CONCUR_SESSION_DISALLWD response, you should retry sending the request after a certain period.

Also, by updating your web services integrations to use TBA, you can take advantage of a more flexible concurrency.

Retrying Failed Web Services Requests

If your client application receives the WS_CONCUR_SESSION_DISALLWD response, you should retry sending the request after a certain period.

Consider the following best practices when implementing your retry logic:

- You should send the first retry attempt after a certain delay. If more attempts are needed, you should increase the delay between attempts.
- Avoid the creation of artificial concurrency peaks due to synchronized retry attempts from different threads or applications. To avoid synchronization, there is no specific recommended length of delay between retry attempts.
- For interactive applications, if retry attempts are needed, you should execute requests asynchronously without need for repeated user action.

Frequently Asked Questions

I currently do not govern my REST concurrency. What happens now?

If you exceed the limit for concurrent requests, an error will be thrown. You need to make sure that your client applications will retry sending requests in the case of an error. This recommendation has always been considered a best practice, and the change in concurrency governance does not introduce any new error codes.

Will my existing integrations break due to these changes?

If the concurrency limits are exceeded, your integration might encounter the error codes listed in Details of the Change. You must modify your integration to handle these exceptions and you must add logic to retry the requests.

Will this affect my sandbox account governance?

Sandbox accounts match your production governance limit. Developer accounts always have a governance limit of five.

Can I monitor my account's concurrency usage level?

You can currently check your account limit on the Web Services Preferences page, under Setup > Integration > Web Services Preferences (Administrator).

I have multiple user accounts being used for concurrent SOAP web services based on request —level authentication. None of them exceed one request at a time. I do not have an SuiteCloud Plus license. Do I now need to buy one?

The total number of SOAP web services concurrent requests across all users will be governed by the account level limit. If you exceed that limit, you might require additional SuiteCloud Plus licenses.

How does this impact the SuiteCloud Plus license?

Prior to version 2017.2, SuiteCloud Plus licenses helped increase the limit on maximum permitted concurrent SOAP requests. This did not have any impact on RESTlets. Starting from version 2017.2, both SOAP and RESTlet requests can benefit from SuiteCloud Plus licenses. SOAP and RESTlet requests both share the same pool for concurrency, which is set at account level. By purchasing one or more SuiteCloud Plus licenses, this limit can be increased.

Is there any impact based on the authentication used for web services? (Request-level or token-based authentication)

For the RESTlet API, there is no impact. All RESTlets are subjected to the account-level governance limit. For SOAP web services, if you want to use request-level authentication, you need to designate a concurrent web services user to enable multiple concurrency. If you switch to token-based authentication (TBA), you do not have to designate anyone as concurrent user. You can run multiple SOAP requests as permitted by the account-level governance limit.

We recommend that you use TBA. This eliminates the need to designate a concurrent web service user as one token can be used in more concurrent requests up to the account limit.

How does the governance change affect the usage of the Concurrent Web Service User option on the user record?

You can continue to use this option to run concurrent SOAP web service requests for designated users. However, these requests will be counted against the account-level governance limit and there is no guarantee or reservation for designated users.

Prior to version 2017.2, if you had one or more SuiteCloud Plus licenses, each designated concurrent web service user is guaranteed 10 concurrent SOAP requests. With the change, if a designated user makes 10 concurrent SOAP web service requests, the number of requests that succeed or fail will depend on other SOAP or REST requests that are being processed.

We recommend that you use token-based authentication (TBA). This eliminates the need to designate a concurrent web service user.

Vocabulary

You should be familiar with the following terms before implementing SuiteTalk integration technology.

- **Type** – A type is an element defined in the XML Schema. See Types for more details.
- **Record Type** – A NetSuite business record type, such as customer, event, or custom record, that has an entry form in the user interface. See Records in Web Services for more details.
- **Search Record Type –** A type that encapsulates the available search criteria for a specific NetSuite business record type. See Records in Web Services for more details.
- **System** – The NetSuite application.
- **Client** – A company with a NetSuite account that is using web services.
- **Requester** – The sender of a Web service request.
- **Write operations** – A web services operation that changes data within a NetSuite account. These include add, addList, update, updateList, delete, and deleteList operations. See Web Services Operations Web Services Operations for information on each operation.
- **Read operations** – A web services operation that retrieves data from a NetSuite account. These include get, getList, getAll, search, searchNext and searchMore operations. See Web Services Operations Web Services Operations for information on each operation.

SuiteTalk Setup

These sections provide the basic information you need to get started with SuiteTalk.

- SuiteTalk Quick Start – provides step-by-step instructions on how to set up your web services environment and start building applications to interface with SuiteTalk. Also provides a path for Downloading Sample Applications provided by NetSuite.
- SuiteTalk Development Considerations – describes development considerations such as account configurations and the enabling of NetSuite features to ensure your web services calls execute successfully.
- Web Services PHP Toolkit – provides setup information for those wanting to use the NetSuite PHP toolkit to develop web services applications.

Before building a web services application to interface with SuiteTalk, the NetSuite web services feature must be enabled in your NetSuite account. See Enabling the Web Services Feature.

After you have enabled web services in your NetSuite account, you must set your web services preferences. See Web Services Preferences.

SuiteTalk Quick Start

This section provides details on using Microsoft .NET or Java to build an application that uses NetSuite web services. This section provides steps for the following tasks:

- Enabling the Web Services Feature
- Enabling the Show Internal IDs Preference
- Building an Application with Microsoft .NET
- Building an Application with Java using Apache Axis
- Downloading Sample Applications
- Capturing SOAP

Note the following:

- Before developing your own SuiteTalk applications, you should see the topic SuiteTalk Development Considerations.
- For information on working with SuiteTalk and PHP, see Web Services PHP Toolkit.
- Although some customers have built SuiteTalk applications using Perl, currently Perl is not officially supported by NetSuite.

Enabling the Web Services Feature

The web services feature must be enabled prior to submitting web services requests.

To enable the web services feature:

1. As administrator, click Setup > Company > Enable Features.
2. Click the **SuiteCloud** subtab.
3. Check the **Web Services** box.

4. Click **Save**.

Note that after you have enabled web services in your NetSuite account, you should set your web services preferences. See Web Services Preferences for more details.

Enabling the Show Internal IDs Preference

NetSuite recommends that you enable the **Show Internal IDs** preference when working with web services or SuiteScript. Enabling this preference lets you see the internal IDs for all fields, records, lists, and custom forms in NetSuite. In web services (and in SuiteScript) you will reference many of these IDs in your code.

To enable the Show Internal IDs preference:

Note: The Show Internal IDs field is only available when at least one of the following features is enabled in your account: Client SuiteScript, Server SuiteScript, SuiteScript Server Pages, SuiteFlow, or Web Services (on the SuiteCloud tab) or Advanced Site Customization or SuiteCommerce Advanced (on the Web Presence tab). For more information, see the help topic Enabling Features.

1. Go to Home > Set Preferences. The **General** subtab is displayed by default.
2. In the **Defaults** section, click **Show Internal IDs**.

When this preference is enabled:

- You can view the internal ID for a field by clicking on that field's label to open the field level help popup window. The internal ID is displayed in the lower right corner of this window.
- You can view the internal ID for a record or a custom field in an Internal ID column that displays on a list or search results page for that type of record or custom field.

Changes to this preference affect only the current user.

For more details, see the help topic Showing Record and Field IDs in Your Account.

- SuiteTalk Platform Overview
- SuiteTalk Quick Start
- SuiteTalk Development Considerations

Building an Application with Microsoft .NET

This section provides details on how to use the Microsoft .NET platform to build a SuiteTalk application. You can also see Downloading Sample Applications for a .NET sample currently available on the NetSuite Developer Portal.

Important: When building an application with Microsoft .NET, NetSuite recommends that you use Visual Studio .NET since it provides an integrated development environment to build and debug your Web service applications. Alternatively, you can download and install the Microsoft .NET Framework SDK. However, this only provides you with the SDK and does NOT provide an integrated IDE.

All code samples in this section use the C# language, however these steps are similar for all other languages supported by Microsoft .NET. These languages include Visual Basic, Visual J#, and C++.

To use Microsoft .NET with NetSuite web services:

1. Install the Microsoft .NET framework.

   Install Microsoft Visual Studio .NET, version 2003 or higher, which includes the .NET framework SDK (RECOMMENDED), or the Microsoft .NET Framework SDK version 1.1 (NOT recommended).
2. Use the .NET framework SDK to automatically generate the client proxy code

   If using Visual Studio .NET (RECOMMENDED):

   1. Start Microsoft Visual Studio .NET.
   2. Create a new project and choose a template, for example, Console Application.
   3. After the project has been created, add a Web reference.

      The Add Web Reference option may be available in the **Project** menu. If not, select **Add Service Reference**, click the **Advanced** button, and click the **Add Web Reference** button.
   4. In the **Add Web Reference** dialog, enter the SuiteTalk WSDL URL and click the **Go** icon.

      A URL for the latest SuiteTalk WSDL is: https://webservices.netsuite.com/wsdl/v2017_2_0/netsuite.wsdl

      Important: The URL for the WSDL may vary from the example shown above. See Effects of Accounts' Host Data Center on URLs for an important note about these URLs.

      Note: Although the material in this guide pertains to the latest NetSuite WSDL, we continue to support endpoints released prior to this version.
   5. Visual Studio inspects the WSDL and displays a summary of the available operations. If security warnings are displayed, click **Yes** as many times as necessary to continue this process.
   6. After the summary is displayed, click the **Add Reference** button to generate the classes.

      When this process is complete, com.netsuite.webservices is listed under **Web References** in the Solution Explorer.
   7. To view all generated proxy classes, do one of the following:
      - Enable the **Show All Files** option in the Project menu
      - (If the above option is not available), right-click the com.netsuite.webservices listing in the Solution Explorer and choose **View in Object Browser**.

   If using only the Microsoft .NET Framework SDK (NOT recommended):

   1. Locate the wsdl.exe file under the Microsoft .NET Framework SDK installation and add it to your system path.
   2. Open a command prompt, and type the following to generate the proxy classes:

      ```
      wsdl /language:<language> <url>
      ```

      Where <language> is your preferred language and <url> is the URL for the NetSuite WSDL.

      For example, generate the proxy classes in C# as follows:

      ```
      C:\project>wsdl /language:cs https://webservices.netsuite.com/wsdl/
      v2017_1_0/netsuite.wsdl
      ```

      A C# file called NetSuiteService.cs is generated.

3. Compile the source code for your proxy into a .NET Assembly using the .NET Framework C# compiler or any other supported compiler.
4. Locate the csc.exe file under the Microsoft .NET Framework SDK installation and add it to your system path.

```
csc.exe /out: NetSuiteService.dll /target:library
reference:system.xml.serialization.dll /
reference:system.web.services.dll
NetSuiteService.cs
```

3. Implement your application by writing your business logic using the generated .NET proxy classes.

   Note that the following code snippets are shown in C#.

   1. Allow the use of objects from the newly created namespace without having to qualify them.

   ```
   using <namespace>.com.netsuite.webservices;
   ```

   2. Instantiate the NetSuite service.

   ```
   NetSuiteService service = new NetSuiteService();
   ```

   3. Enable support for multiple cookie management.

   ```
   service.CookieContainer = new CookieContainer();
   ```

   4. Create a valid session by populating the Passport object.

   ```
   //invoke the login operation
   Passport passport = new Passport();
   passport.account = "TSTDRV96";
   passport.email = "username@netsuite.com";
   RecordRef role = new RecordRef();
   role.internalId = "3";
   passport.role = role;
   passport.password = "mypassword";
   Status status = service.login( passport ).status;
   ```

   5. Implement your business logic. For example, create a new customer in NetSuite.

   ```
   Customer cust = new Customer();
   cust.entityID( “XYZ Inc” );
   WriteResponse response = service.add( cust );
   ```

   6. Logout to invalidate the current session.

```
service.logout();
```

Building an Application with Java using Apache Axis

This section provides details on how to use the Apache Axis framework (versions 1.3 and 1.4) to build a SuiteTalk application.

Note: The Java sample application available on the Developer portal provides an in-depth look at how to build an application using Apache Axis and NetSuite web services. See Downloading Sample Applications for more information.

To use the Apache Axis framework with NetSuite web services:

1. Install the Java 2 Platform.

   Download and install the Java 2 Platform, Standard Edition, version 1.4 or higher from http://java.sun.com/j2se/1.4.2/download.html. Ensure that the executables are available through the system path.

2. Install Apache Axis.

   1. Download and install Apache Axis from http://ws.apache.org/axis/.

      Important: Only Axis versions 1.3 and 1.4 are supported in NetSuite web services.

   2. Download and install the Apache Axis patch for cookie management from NetSuite: http://www.netsuite.com/portal/developers/resources/suitetalk-sample-applications.shtml

      The NetSuite Web service implementation requires the client application to support multiple cookies on one line, as is the standard for cookies. There is a bug in Apache Axis that puts each cookie on its own line in the HTTP Headers. The patch version of the axis.jar fixes this problem. After it is downloaded, replace the existing axis.jar file in the lib directory of your Axis root directory with this version.

   3. After it is installed, set an environment variable called AXIS_HOME to point to the Axis installation directory.

3. Install Apache Ant (Optional).

   Download and install Apache Ant, version 1.5 or higher, from http://ant.apache.org/. Apache Ant is a Java-based build tool that facilitates automation of the build process, including generating the proxy classes from the NetSuite WSDL.

   See the build.xml file in the Java sample application for a complete Ant build script.

4. Configure Java to generate unreferenced types.

   Set the **all** parameter in your axis-wsdl2java ant task to **true**. For example:

```
<axis-wsdl2java timeout="120000" output="${generated.src.dir}" verbose="true"
url="${wsdl-1.3.url}" all="true" wrapArrays="true">
```

5. Use Apache Ant to automatically generate the client proxy code.

   Using Apache Axis from the command line

   Use the WSDL2Java utility to generate the proxy classes as follows:

```
java -cp <classpath> org.apache.axis.wsdl.WSDL2Java <url>
```

   Where the <classpath> points to the appropriate Apache Axis JAR files and <url> is the URL for the NetSuite WSDL.

   For example, the following commands will set the class path and generate the proxy classes:

```
>set
CP=%AXIS_HOME%\lib\axis.jar;%AXIS_HOME%\lib\jaxrpc.jar;%
```

```
AXIS_HOME%\lib\commons-discovery.jar;%AXIS_HOME%\lib\wsdl4j.jar;
%AXIS_HOME%\lib\saaj.jar;>java -cp %CP% org.apache.axis.wsdl.WSDL2Java
https://webservices.netsuite.com/wsdl/v2017_1_0/netsuite.wsdl
```

Important: The URL for the WSDL may vary from the example shown above. See Effects of Accounts' Host Data Center on URLs for an important note about these URLs.

Using the Apache Axis task in Apache Ant

Create a build target that uses the WSDL2Java Ant task as follows where the ${generated.src.dir} variable is the directory where the source code is generated and the ${wsdl.url}variable points to the NetSuite WSDL.

```
<target name="generate.interfaces" description="Generates
client interfaces using Axis">
<echo>Generating client interfaces using Apache Axis</echo>
<axis-wsdl2java output="${generated.src.dir}" verbose="true"
url="${wsdl.url}">
<mapping namespace="http://axis.apache.org/ns/interop"
package="interop"></mapping>
</axis-wsdl2java>
</target>
```

6. Ensure that the Ant executables are available on the system path. Run the Ant task as follows:

```
ant generate.interfaces
```

7. Implement your application by writing your business logic using the generated axis proxy classes.
    1. Locate the NetSuite service.

    ```
    NetSuiteServiceLocator service = new NetSuiteServiceLocator();
    ```

    2. Enable support for multiple cookie management.

    ```
    service.setMaintainSession( true );
    ```

    3. Get the NetSuite port.

    ```
    NetSuitePortType port = service.getNetSuitePort();
    ```

    4. Create a valid session by populating the Passport object and then invoking the login operation.

    ```
    passport.setEmail( "username@netsuite.com" );
    passport.setPassword( "mypassword" );
    role.setid( "3" );
    passport.setRole( role );
    passport.setAccount( "TSTDRV96" );
    Status status = port.login( passport ).getStatus();
    ```

    5. Implement your business logic. For example, create a new customer in NetSuite.

    ```
    Customer cust = new Customer();
    ```

```
            cust.setEntityID( “XYZ Inc” );
            WriteResponse response = port.add( cust );
```

6. Logout to invalidate the current session.

```
            port.logout();
```

Downloading Sample Applications

Web services developers can download sample applications from the NetSuite Developer Portal, at http://www.netsuite.com/portal/developers/resources/suitetalk-sample-applications.shtml. CRM and ERP sample applications are provided in both Java and .NET.

Capturing SOAP

When writing web services you will want to capture and analyze SOAP request and response messages associated with an operation call. You can use such products as Actional Diagnostics or SOAPScope. You can also review your SOAP requests and responses from the pages where they are available in NetSuite. You can download these files from the Execution Log of each integration record, or you can use the Web Services Usage Log, which shows the logs from all integration records. For details on working with integration records, see Integration Management. For details on using the usage log, see Using the Web Services Usage Log.

SuiteTalk Development Considerations

This section covers the following topics:

- Development Considerations Overview
- NetSuite Features in Web Services
- Effects of Account Configuration in Web Services
- Effects of Accounts' Host Data Center on URLs
- Enumerations and Special Characters

Development Considerations Overview

Before you begin writing SuiteTalk applications, you should be aware that SuiteTalk adheres to the same role-based permission structure enforced in the NetSuite UI. Because a SuiteTalk application needs a pair of sign-in credentials to login, its permission to various operations and records is subjected to the role it uses, like it would when it is used for a browser session. For example, a SuiteTalk application that logs in with the Accountant role will receive the same permissions as it will logging in to the browser interface using the same role.

Note: See Roles and Permissions in Web Services for additional details.

Another characteristic of SuiteTalk is that its behavior is very similar to that of the NetSuite UI. The workflow of a SuiteTalk application and its underlying SOAP exchange with NetSuite tightly mimics the browser interface. For example:

1. A successful login operation must be performed to obtain a session to perform any subsequent operations. (See the documentation on the login operation and on web services Authentication for Web Services for more details.)

2. Some add operations are done in a tandem, such as loading an existing sales order to gain the context and then adding a new item fulfillment record. The loading of the sales order (using the initialize API) is the first operation; adding the item fulfillment record is the second.
3. Restrictions and requirements on custom forms are honored in SuiteTalk. Therefore, a SuiteTalk application's attempt to set a field that is hidden on a form results in a permission error. Likewise, as in the UI, a SuiteTalk application must set fields that are required on the form it is using.

The combination of permission adherence and similar behavioral patterns between SuiteTalk and the NetSuite UI provides a consistent and predictable platform for developers.

Important: When you are unsure of how to achieve something with SuiteTalk, try observing how it is done in the UI, then replicate it programmatically.

Note: A SuiteTalk application should use the response object to handle any errors, if any, generated by a Web service operation.

Important: SuiteTalk uses the HTTP protocol, therefore service interruptions can occur from time to time. If you receive an HTTP 503 error (Service Unavailable), try sending your web services request later.

NetSuite Features in Web Services

The NetSuite UI lets you enable or disable certain features. When designing your web services, it is important to know which features must be enabled in order for the service to execute properly. If the service calls for a function that is NOT available because the associated feature is disabled, a SOAP **InsufficientPermissionFault** occurs causing the entire service to fail. For example, in the NetSuite UI you can enable or disable the Opportunities feature for CRM. If disabled, a Web service call to add an OpportunityItem will fail.

Web services is itself a feature that **must** be enabled prior to submitting web services requests. To enable the feature, see Enabling the Web Services Feature.

Effects of Account Configuration in Web Services

Form customization in an account or enabling or disabling features may result in required fields being added to various forms used on records. For each web services request, fields available for the specified record are validated against the data being submitted, and errors are returned where field validation fails. However, for get, add, addList and search requests, it is possible that the field requirements change mid-request, resulting in errors for a **subset** of the request.

For example, suppose you want to submit 20 new customer records using the addList operation. Upon submission, field validation passes for the request. However, after the first 15 customers are added, the required fields may be changed within your NetSuite account, causing an error to be returned for the 5 remaining items.

Note: For more specific information on working with NetSuite records, fields, and forms in web services, see Records, Fields, Forms, and Sublists in Web Services.

Effects of Accounts' Host Data Center on URLs

NetSuite hosts customer accounts in multiple data centers. For that reason, the correct URL for web services access varies depending on the data center hosting the account. For example, the URL could be webservices.netsuite.com, webservices.na1.netsuite.com, or another variant. Your integration **must** incorporate logic that dynamically determines the correct URL.

Note: As of 2017.2, account-specific domains are supported for web services, and you can access your web services domain at the following URL, where 123456 is your account ID: 123456.suitetalk.api.netsuite.com. The data center-specific domains supported before 2017.2 will continue to be supported. For more information, see the help topic URLs for Account-Specific Domains.

The following methods exist for determining the correct URL:

- In the 2012.2 and later endpoints, you can use the getDataCenterUrls operation. This operation is not supported in earlier endpoints.
- You can use the REST roles service provided by NetSuite.
  - For general information about this service, see the help topic Using the REST roles Service to Get User Accounts, Roles, and Domains.
  - For example code that uses this REST service, see:
    - Java REST Example for Web Services Domain Discovery
    - .NET REST Example for Web Services Domain Discovery

    The NetSuite Developer Resources sample applications also incorporate RESTful code for dynamic discovery of web services domain. These samples are available at: http://www.netsuite.com/portal/developers/resources/suitetalk-sample-applications.shtml.

Note: Starting from version 2017.2, you can also use the DataCenterUrls REST Service to dynamically discover the correct URLs

Important: When used in a production account, your web services integration **must** incorporate logic that dynamically determines the correct URL for web services access.

Web Services URLs for Sandbox, EU Sandbox, and Release Preview Environments

For information on the correct URL to use for web services access to sandbox, EU sandbox, and Release Preview accounts, see the help topic Understanding NetSuite URLs and Data Centers.

Java REST Example for Web Services Domain Discovery

This example includes a sample call to the REST roles service to dynamically discover the correct web services domain and a sample client.

Important: NetSuite hosts customer accounts in multiple data centers. For that reason, the correct URL for web services access varies depending on the data center hosting the account. Your integration **must** incorporate logic that dynamically determines the correct URL, either by using the REST roles service, as described in this topic, or another approach. In general, with the 2012.2 and later endpoints, you should use the getDataCenterUrls operation. The REST roles service is intended for integrations that use the 2012.1 or an earlier endpoint.

As of 2017.2, account-specific domains are supported for web services, and you can access your web services domain at the following URL, where 123456 is your account ID: 123456.suitetalk.api.netsuite.com. The data center-specific domains supported before 2017.2 will continue to be supported. For more information, see the help topic URLs for Account-Specific Domains.

Sample Java Call to the REST roles Service

```
package com.netsuite.prototypes;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;
import javax.net.ssl.HttpsURLConnection;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
public class GetRESTDomain
{
   static class URLTriplet
   {
      String restDomain;
      String systemDomain;
      String webservicesDomain;
   }
   private BufferedReader _br = null;
   public GetRESTDomain()
   {
      _br = new BufferedReader(new InputStreamReader(System.in));
   }
   /** * Retrieve domains of a NetSuite account through RESTful WS call.***/

   public URLTriplet getDataCenterUrls(String nsAccount, String nsEmail, String nsPassword) thr
ows IOException, ParseException
   {
      String sysURL        = "https://rest.netsuite.com/rest/roles";
      URLTriplet urls      = new URLTriplet();
      // 'nlauth_account' should NOT be specified for this type of request otherwise an error i
s returned.//
      String nlAuth = "NLAuth nlauth_email=" + nsEmail + ", nlauth_signature=" + nsPassword;
      URL u = new URL(sysURL);
      HttpsURLConnection uc = (HttpsURLConnection) u.openConnection();
      uc.setAllowUserInteraction(Boolean.FALSE);
      uc.setInstanceFollowRedirects(Boolean.FALSE);
      uc.setRequestMethod("GET");
      uc.setRequestProperty("Authorization", nlAuth);
      BufferedReader in = new BufferedReader(new InputStreamReader(uc.getInputStream()));
      String inputLine = in.readLine();
      in.close();
      if (inputLine.length() > 0)
      {
         JSONArray jsonArr = (JSONArray) new JSONParser().parse(inputLine);

         for (int i=0 ; i<jsonArr.size() ; i++)
         {
            JSONObject json = (JSONObject) jsonArr.get(i);
            // Validate JSON object if (json.containsKey("account") && json.containsKey("dataCe
nterURLs"))
            {
               JSONObject jsonAccount     = (JSONObject) json.get("account");
               JSONObject jsonDataCenters = (JSONObject) json.get("dataCenterURLs");
               // Validate account and retrieve domains if (jsonAccount.get("internalId").toStr
ing().equalsIgnoreCase(nsAccount))
               {
                  urls.restDomain = jsonDataCenters.get("restDomain").toString();
                  urls.webservicesDomain = jsonDataCenters.get("webservicesDomain").toString();

                  urls.systemDomain = jsonDataCenters.get("systemDomain").toString();
                  break;
               }
            }
         }
      }

      return urls;
   }

   public static void main(String[] args) throws IOException
{
      GetRESTDomain ns = new GetRESTDomain();
      while (true)
   {
         System.out.println("GET REST DOMAIN URL");
         System.out.print("Account      : ");
         String account = ns._br.readLine().trim();

         if (account.isEmpty())
         {
            break;
         }

         System.out.print("Email address: ");
         String email = ns._br.readLine().trim();
         System.out.print("Password     : ");
         String password = ns._br.readLine().trim();
         try
         {
            URLTriplet urls = ns.getDataCenterUrls(account, email, password);
            System.out.println("\nREST domain URL        : " + urls.restDomain + "\n");
            System.out.println("\nWeb services domain URL: " + urls.webservicesDomain + "\n");
            System.out.println("\nSystem domain URL      : " + urls.systemDomain + "\n");
         }
catch (IOException e)
         {
            e.printStackTrace();
         }
catch (ParseException e)
         {
            e.printStackTrace();
         }
      }
      System.out.print("\nPress any key to exit...");
      ns._br.readLine().trim();
   }
}
```

Sample AXIS Client

```
package com.netsuite.prototypes;
import com.netsuite.devtools.lists.employees.Employee;
import com.netsuite.devtools.platform.NetSuitePortType;
import com.netsuite.devtools.platform.NetSuiteServiceLocator;
import com.netsuite.devtools.platform.core.RecordRef;
import com.netsuite.devtools.platform.core.types.RecordType;
/*** Example showing how to get web services domain pointing to the account’s data center throu
gh RESTful WS call for endpoints older than 12.2. (The method getDataCenterUrls() was introduce
d in the 12.2 endpoint). ***/
public class AXISClient
{
   private static final String ENDPOINT_VERSION = "NetSuitePort_2017_1";
   private static final String ACCOUNT = "123456";
   private static final String PASSWORD = "123456@netsuite.com";
   private static final String EMAIL = "passwd";
   NetSuitePortType _port;
   public AXISClient(String acct, String email, String passwd) throws Exception
   {
      NetSuiteServiceLocator service = new NetSuiteServiceLocator();
      GetRESTDomain restHelper = new GetRESTDomain();
      String wsDomain = restHelper.getDataCenterUrls(acct, email, passwd).webservicesDomain;
      service.setNetSuitePortEndpointAddress(wsDomain + "/services/" + ENDPOINT_VERSION);
      _port = service.getNetSuitePort();
      // initialize _port - authentication// ...
   }
   public static void main(String[] args) throws Exception
   {
      AXISClient client = new AXISClient(ACCOUNT, EMAIL, PASSWORD);
      // _port points to the correct data center. Perform operations as usual.// ...
      RecordRef rr = new RecordRef();
      rr.setInternalId("25");
      rr.setType(RecordType.employee);
      Employee e = (Employee)client._port.get(rr).getRecord();
      System.out.println(e.getFirstName() + " " + e.getLastName());
   }
}
```

.NET REST Example for Web Services Domain Discovery

This example includes a sample call to the REST roles service to dynamically discover the correct web services domain and a sample client.

Important: NetSuite hosts customer accounts in multiple data centers. For that reason, the correct URL for web services access varies depending on the data center hosting the account. Your integration **must** incorporate logic that dynamically determines the correct URL, either by using the REST roles service, as described in this topic, or another approach. In general, with the 2012.2 and later endpoints, you should use the getDataCenterUrls operation. The REST roles service is intended for integrations that use the 2012.1 or an earlier endpoint.

As of 2017.2, account-specific domains are supported for web services, and you can access your web services domain at the following URL, where 123456 is your account ID: 123456.suitetalk.api.netsuite.com. The data center-specific domains supported before 2017.2 will continue to be supported. For more information, see the help topic URLs for Account-Specific Domains.

Sample .NET Call to the REST roles Service

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Collections.Generic;
using System.Xml.Linq;
using System.Linq;
using System.Runtime.Serialization.Json;
using System.Net;
using System.IO;
using System.Xml;

namespace NSClientCRM
{
    class GetRESTDomain
    {
        public URLTriplet getDataCenterUrls(String nsAccount, String nsEmail, String nsPassword
)
        {
            // Preset Values
            string sysURL = "https://rest.netsuite.com/rest/roles";
            URLTriplet urls = new URLTriplet();

            HttpWebRequest httpWebRequest = (HttpWebRequest)HttpWebRequest.Create(sysURL);

            // 'nlauth_account' should NOT be specified for this type of request otherwise an e
rror is returned.
            // Issue 238527
            httpWebRequest.Headers.Add("Authorization",
                                       "NLAuth nlauth_email=" + nsEmail + "," +
                                       "nlauth_signature=" + nsPassword);

            // set content type
            httpWebRequest.ContentType = "application/json";

            // execute http GET
            WebResponse webResponse = httpWebRequest.GetResponse();
            Stream httpBody = webResponse.GetResponseStream();
            string jsonContent = "";
            int i = 0;

            StreamReader streamReader = new StreamReader(httpBody);

            while (jsonContent != null)
            {
                i++;
                jsonContent = streamReader.ReadLine();
                Stream mainContent = streamReader.BaseStream;

                if (jsonContent != null)
                {

                    // convert json content to bytes
                    byte[] bytesJsonContent = Encoding.Unicode.GetBytes(jsonContent);

                    // convert the json to an xml format so that xpath can be used to locate sp
ecific
                    // nodes & attributes
                    XmlDictionaryReader xmlReader =
                        JsonReaderWriterFactory.CreateJsonReader
                        (bytesJsonContent, new XmlDictionaryReaderQuotas());

                    // this returns the top-level element of the xml document;
                    // all of the following xpath statements are executed off of this element
                    XElement root = XElement.Load(xmlReader);

                    //Linq Query
                    IEnumerable<XElement> restDomains =
                    from c in root.Elements("item")
                    where (string)c.Element("account").Element("internalId").Value == nsAccount

                    select c;


                    // Outputs Data Center URLs for the specified account
                    if (restDomains != null)
                    {
                        foreach (XElement elementFromLinq in restDomains)
                        {
                            string accountIntId = elementFromLinq.Element("account").Element("i
nternalId").Value;
                            string accountName = elementFromLinq.Element("account").Element("na
me").Value;
                            Console.WriteLine("Data Center URLs for " + accountIntId + " " + ac
countName);
                            Console.WriteLine("-----------------------");
                            Console.WriteLine("Role: " + elementFromLinq.Element("role").Elemen
t("name").Value);
                            Console.WriteLine("Rest Domain: " + elementFromLinq.Element("dataCe
nterURLs").Element("restDomain").Value);
                            Console.WriteLine("System Domain: " + elementFromLinq.Element("data
CenterURLs").Element("systemDomain").Value);
                            Console.WriteLine("Web Services Domain: " + elementFromLinq.Element
("dataCenterURLs").Element("webservicesDomain").Value);
                            Console.WriteLine(" ");
                            urls.webservicesDomain = elementFromLinq.Element("dataCenterURLs").
Element("webservicesDomain").Value;
                            urls.systemDomain = elementFromLinq.Element("dataCenterURLs").Eleme
nt("systemDomain").Value;
                            urls.restDomain = elementFromLinq.Element("dataCenterURLs").Element
("restDomain").Value;
                            break;
                        }
                    }

                    Console.Write("Number of Roles for this Account: " + restDomains.Count());
                    if (restDomains != null)
                    {
                        Console.WriteLine("Roles");
                        Console.WriteLine("-----------------------");
                        foreach (XElement elementFromLinq in restDomains)
                        {
                            Console.WriteLine(elementFromLinq.Element("role").Element("name").V
alue);
                        }

                    }
                }
            }
            return urls;
        }
    }
}
```

Sample .NET Client

```
//...
NSClientCRM ns = new NSClientCRM();

GetRESTDomain restHelper = new GetRESTDomain();
URLTriplet urls = restHelper.getDataCenterUrls("123456", "123456@netsuite.com", "passwd");

ns._service.Url = urls.webservicesDomain + "/services/NetSuitePort_2017_1";
//...
```

Enumerations and Special Characters

When enumerations contain either special characters (“(“) or reserved keywords (“private” or “public” for example), both .NET and Axis may generate less usable code on the client side. To alleviate this problem, all enumerated values in NetSuite web services are prefixed with an underscore “_”, except for enumerated values from the platformCore, platformCoreTyp, platformFaults, platformFaultsTyp, and platformMsgs XSDs. For example, without the “_”, .NET prepends an “@” symbol to the variable, as in “@private”.

Image References in Web Services

To reference an image, the image must first be uploaded to the NetSuite file cabinet, and then referenced using the image name specified in the file cabinet. For more information on working with images (and other file types) in the file cabinet, see the help topic File.

Web Services Preferences

NetSuite preferences control how certain services are executed. Therefore, before using web services with your NetSuite account, ensure that the appropriate preferences are enabled, as defined in the following sections:

- Company-Wide Preferences
- One-Time Preferences
- Resetting Default Behavior
- Setting the Internal ID Preference
- Caching Behavior in Web Services

Company-Wide Preferences

In some cases, you may want to set company-wide preferences that affect all web services calls made to your NetSuite account. Changes to these preferences are propagated to every user.

You can set these preferences at Setup > Integration > Web Services Preferences.

Some of these preferences can be overridden when you send individual web services requests. For details, see One-Time Preferences. Be aware also that additional preferences are available at the request level.

Note: The Web Services Preferences page also includes information whether the web services and RESTlet concurrency governance is enabled for your account, and about the available concurrency limit for your account. For information about web services and RESTlet concurrency governance, see Web Services and RESTlet Concurrency Governance Starting From Version 2017.2

The following preferences can be set on a company-wide basis:

- Disable Mandatory Custom Field Validation
- Run Server SuiteScript and Trigger Workflows
- Require Approval During Auto-Installation of Integration
- Search Page Size
- Treat Warnings as Errors
- Use Conditional Defaults on Add
- Use Conditional Defaults on Update

Disable Mandatory Custom Field Validation

This preference affects the handling of custom fields that are configured in the UI to be mandatory. If you set this preference to true, the fields are **not** treated as required during web services requests. If the preference is set to false, the fields are required. In these cases, if you fail to provide the required value, the system returns a USER_ERROR, with instructions to provide a value for the field.

When deciding whether to enable this preference, consider whether the data being provided comes from a system that has equivalent fields to the custom field.

This preference can be overridden at the request level. For details, see disableMandatoryCustomFieldValidation.

Run Server SuiteScript and Trigger Workflows

This preference setting is applied to the entire company. However, you can overwrite this preference in each web services request. For details, see Request-Level Preferences.

This preference enables server SuiteScripts and triggers workflows and is selected by default. If this preference is not selected, all SuiteScript scripts and workflows are disabled.

If you disable this option, you do both of the following:

- Disable server SuiteScripts that otherwise would have been triggered by web services requests.
- Disable workflows that otherwise would have been triggered by web services requests.

If you are bringing historical data into NetSuite, consider disabling this option to increase speed of operations and block executions of additional logic performed automatically by executed scripts and workflows. If you are syncing live data or running a partner application (for example, Outlook Sync) it is recommended that you use the default option, Server. In other words, you should consider enabling server SuiteScript to ensure that your business logic runs for your integrated application. However, note that doing so may have a negative performance impact.

Important: To ensure that your business logic is fully executed, always use the default option, Server. Additionally, changing the default option can potentially break your working integration. You are discouraged from doing so unless you are an advanced web services user.

Require Approval During Auto-Installation of Integration

Auto-installation is the automatic creation of an integration record in your NetSuite account. For example, in some cases, a partner may provide you with an updated version of an application. This update may include a new application ID created by using an integration record. The first time you send a request using this application, a new integration record is created in your account.

The **Require Approval during Auto-Installation of Integration** preference affects whether this new record is automatically enabled. If the preference is set to false, the State field on the new application is automatically set to Enabled, and all requests are permitted. If the preference is set to true, the State field on the new integration record is set to Waiting for Approval. In the latter case, you must manually edit the record and set the State to Enabled. Until you do, all requests sent by that application are blocked.

Search Page Size

The Search Page Size preference determines the number of records returned in the results of a search. This preference affects both synchronous and asynchronous searches. When you set this preference on a company-wide basis, valid values are between 5 and 1,000. The default value is 1,000.

This preference can be overridden at the individual search level. When setting this preference for a particular search, note that the maximum for an asynchronous search is greater than the maximum that can be set at the company-wide level. For more details, see pageSize.

Treat Warnings as Errors

When this preference is enabled, warning messages generated by NetSuite are treated as errors that cause an exception, resulting in rejection of the request. For more information on the difference between errors and warnings, see Web Services Warnings, Errors, and Faults.

This preference can be overridden at the request level. For details, see warningAsError.

Use Conditional Defaults on Add

This preference is available only for the 2009.2 endpoint and earlier. Note that these older endpoints are no longer supported. For details, see Support Has Ended for SuiteTalk 2009.2 and Earlier Endpoints.

For details on this preference, see Resetting Default Behavior.

This preference can be overridden at the request level. For details, see useConditionalDefaultsOnAdd.

Use Conditional Defaults on Update

This preference is available only for the 2009.2 endpoint and earlier. Note that these older endpoints are no longer supported. For details, see Support Has Ended for SuiteTalk 2009.2 and Earlier Endpoints.

For details on this preference, see Resetting Default Behavior.

This preference can be overridden at the request level. For details, see useConditionalDefaultsOnUpdate.

One-Time Preferences

If appropriate, you can set preferences at the request level. These settings override preferences that might have been established for your NetSuite account as a whole.

Note that some preferences exist only at the company-wide level and therefore cannot be overridden. Conversely, some preferences exist only at the request level and therefore can be set only during a specific request or search. That is, they have no company-wide equivalent.

The available request-level preferences are described in the Request-Level Preferences topic.

For details on company-wide preferences, see Company-Wide Preferences.

Request-Level Preferences

Certain web services preferences can be set at the request level. When you use this approach, the request-level preferences override any company-wide preferences that might have been set.

Preferences available at the request level are part of the Preferences complex type. This type is defined in the platform messages XSD file. You can also view the Preferences complex type on the SuiteTalk Schema Browser’s Preferences reference page.

Note that all of these preferences take boolean values.

The available preferences are:

- disableMandatoryCustomFieldValidation
- disableSystemNotesForCustomFields
- ignoreReadOnlyFields
- runServerSuiteScriptAndWorkflowTriggers
- useConditionalDefaultsOnAdd
- useConditionalDefaultsOnUpdate
- warningAsError
- bodyFieldsOnly
- pageSize

- returnSearchColumns

Note: When setting a preference value, you must use camelCase style capitalization for the preference type name. If you do not, the system does not return an error, but your settings are ignored.

disableMandatoryCustomFieldValidation

This preference affects the handling of custom fields that are configured in the UI to be mandatory. If you set this preference to true, the fields are **not** treated as required during web services requests. If the preference is set to false, the fields are required. In these cases, if you fail to provide the required value, the system returns a USER_ERROR, with instructions to provide a value for the field.

When deciding whether to enable this preference, consider whether the data being provided comes from a system that has equivalent fields to the custom field.

This preference can also be set at the company-wide level. It displays in the UI as Disable Mandatory Custom Field Validation. If you do not set this value at the request level, NetSuite uses the company-wide preference.

disableSystemNotesForCustomFields

System notes are system-generated records that track changes to a record, including changes to specific field values. This preference lets you disable the creation of system notes for changes to custom fields. Depending on your integration, using this preference may increase performance.

This preference is available at the request level only.

Important: System notes are important for maintaining a complete audit trail. Therefore, if a custom field contains sensitive information that is critical for audit purposes, you should carefully consider whether or not to enable this preference. For more details about system notes, see the help topic Auditing Data Changes using Searches.

ignoreReadOnlyFields

This preference affects the way the system behaves when you mistakenly submit a value for a read-only field. If you set this preference to true, the system ignores these mistakes.

It is recommended that you set this preference to true in the following situations:

- If you are getting a record, making a change to it, and then submitting it.
- If you are initializing a record.

In these situations, if you set the preference to true, you do not have to take the time to remove values for read-only fields. Otherwise, if values for these fields exist, the system returns an error reading INSUFFICIENT_PERMISSION.

This preference is available at the request level only.

runServerSuiteScriptAndWorkflowTriggers

By using the **runServerSuiteScriptAndTriggerWorkflows** preference, you can control SuiteScript and trigger workflows per request. If you do not include this preference in a request, the company preference set on the Web Services Preferences page is used. If you include this preference in the request, it overrides the company preference set in the UI.

The preference is set in the request in the following way:

```
<runServerSuiteScriptAndTriggerWorkflows>true</runServerSuiteScriptAndTriggerWorkflows>
```

The value of the preference is set corresponding to the value from the request. If the preference is not included in the request, the UI preference is used. If the preference is included in the request, it overwrites the company preference set in the UI. This preference can be set on the UI using the **Run SuiteScript and Workflow Triggers** check box.

If you disable this option, you do both of the following:

- Disable server SuiteScripts that otherwise would have been triggered by web services requests.
- Disable workflows that otherwise would have been triggered by web services requests.

If you are bringing historical data into NetSuite, consider disabling this option to increase speed of operations and block executions of additional logic performed automatically by executed scripts and workflows. If you are syncing live data or running a partner application (for example, Outlook Sync) it is recommended that you use the default option, Server. In other words, you should consider enabling server SuiteScript to ensure that your business logic runs for your integrated application. However, note that doing so may have a negative performance impact.

Important: To ensure that your business logic is fully executed, always use the default option to run SuiteScript and Server Triggers. Additionally, changing the default option can potentially break your working integration. You are discouraged from doing so unless you are an advanced web services user.

The Control SuiteScript and Workflow Triggers in Web Services Request permission is related to the preference for disabling scripts and workflow triggers per request. This permission is a Setup type permission. A user must have the Full level of this permission to set the new SOAP header preference for disabling scripts and workflow triggers per request. If a user with insufficient permissions sends a request with this preference set, the request fails with the following error:

```
<platformFaults:code>INSUFFICIENT_PERMISSION</platformFaults:code>
<platformFaults:message>You do not have permission to control SuiteScript and Workflow Triggers
 in Web Services Request.</platformFaults:message>
```

The global preference applies to requests sent by users who do not have permission to set a request-level preference.

useConditionalDefaultsOnAdd

This preference is available only for the 2009.2 endpoint and earlier. Note that these older endpoints are no longer supported. For details, see Support Has Ended for SuiteTalk 2009.2 and Earlier Endpoints.

For details on the behavior of the preference, see Resetting Default Behavior.

This preference can also be set at the company-wide level. It displays in the UI as Use Conditional Defaults on Add. If you do not set this value at the request level, NetSuite uses the company-wide preference.

useConditionalDefaultsOnUpdate

This preference is available only for the 2009.2 endpoint and earlier. Note that these older endpoints are no longer supported. For details, see Support Has Ended for SuiteTalk 2009.2 and Earlier Endpoints.

For details on the behavior of the preference, see Resetting Default Behavior.

This preference can also be set at the company-wide level. It displays in the UI as Use Conditional Defaults on Update. If you do not set this value at the request level, NetSuite uses the company-wide preference.

warningAsError

When this preference is enabled, warning messages generated by NetSuite are treated as errors that cause an exception, resulting in rejection of the request. For more information on the difference between errors and warnings, see Web Services Warnings, Errors, and Faults.

This preference can also be set at the company-wide level. It displays in the UI as Treat Warnings as Errors. If you do not set this value at the request level, NetSuite uses the company-wide preference.

bodyFieldsOnly

The bodyFieldsOnly preference is boolean. You use it to specify whether or not sublist values are returned in search results. In other words, if you set this value to true, only body fields are returned. If you set it to false, sublist values are also returned.

Leaving this preference set to its default value, which is true, can significantly improve performance.

Note that in an advanced search, this preference is ignored.

Note: The bodyFieldsOnly request-level preference affects the way formulas on forms are processed. As a result, if the values of custom fields are calculated using formulas, the search might return incorrect results. If the calculated values of formulas are incorrect, set the bodyFieldsOnly preference to false to make sure the correct values are returned.

pageSize

The pageSize preference takes an integer. You use this preference to specify the number of records returned on a single page of search results. If more results exist, you can subsequently use the searchMore operation to retrieve each additional page of results. The searchMore operation uses the same page-size preference that you originally set.

When you set a value for pageSize, the following limits apply.

Search Type	Minimum	Maximum
Synchronous	5	1,000
Asynchronous	5	2,000

If you enter an invalid number, the system may return an error code reading WS_INVALID_SEARCH_PAGE_SIZE.

If you do not set a value for this preference in your search, the system uses the value that has been set for the entire system. For more details, see Search Page Size.

returnSearchColumns

The returnSearchColumns preference is relevant when you are using an advanced search to execute an existing saved search. In these cases, you might want the search to return full records, as opposed to columns. To accomplish this, you would set the preference to false.

The default value for the preference is true. Note that if you set this preference to true and you do not specify search return columns, the system returns an error. Similarly, if you are not executing a saved search and you set the preference to false, the system returns an error.

For more information on advanced searches, see Advanced Searches in Web Services.

Sample Code

The following examples show how to set request- and search-level preferences.

For details on each preference, see Request-Level Preferences.

Note: When setting a preference value, you must use camelCase style capitalization for the preference type name. Although no errors are thrown if this is not used correctly, your settings will be ignored.

C#

In the following example, a general search-level preference is set.

```
// Set up request-level preferences as a SOAP header

Preferences prefs = new Preferences();
_service.preferences = prefs;

// Set preference to specify that warnings should be treated as errors

prefs.warningAsErrorSpecified = true;
prefs.warningAsError = false;
```

In this example, search preferences are set.

```
// Set search preference and invoke search operation

_service.searchPreferences.pageSize = 20;
_service.searchPreferences.pageSizeSpecified = true;
SearchResult response = _service.search( custSearch );
```

Java

In the following example, several request-level preferences are set.

```
private void setPreferences() throws SOAPException
{
   stub = (NetSuiteBindingStub) _port;
   stub.clearHeaders();
   SOAPHeaderElement prefHeader =
   new SOAPHeaderElement("urn:messages_2017_1.platform.webservices.netsuite.com", "preferences"
);
   Preferences prefs = new Preferences();
```

```
    prefs.setWarningAsError(new Boolean(false));
    prefs.setUseConditionalDefaultsOnAdd(new Boolean(false));
    prefs.setUseConditionalDefaultsOnUpdate(new Boolean(false));
    prefs.setDisableMandatoryCustomFieldValidation(new Boolean(true));
    prefs.setDisableSystemNotesForCustomFields(new Boolean(true));
    prefs.setIgnoreReadOnlyFields(new Boolean(true));
    prefHeader.setObjectValue(prefs);

    stub.setHeader(prefHeader);
}
```

In the following example, seach-level preferences are set.

```
NetSuiteBindingStub stub = (NetSuiteBindingStub)aPort;
stub.clearHeaders();
SOAPHeaderElement searchPrefHeader = new
SOAPHeaderElement("urn:messages_2017_1.platform.webservices.netsuite.com","searchPreferences");

SearchPreferences searchPrefs = new SearchPreferences();
searchPrefs.setPageSize(new Integer(nPageSize));
searchPrefs.setBodyFieldsOnly(isBodyFieldsOnly);
searchPrefHeader.setObjectValue(searchPrefs);
stub.setHeader(searchPrefHeader);
```

SOAP Request

The following SOAP request shows several request-level preferences being set.

```
<soap:Header>
<platformMsgs:preferences>
   <platformMsgs:warningAsError>true</platformMsgs:warningAsError>
   <platformMsgs:useConditionalDefaultsOnAdd>true</platformMsgs:useConditionalDefaultsOnAdd>
   <platformMsgs:useConditionalDefaultsOnUpdate>true</platformMsgs:useConditionalDefaultsOnUpda
te>
   <platformMsgs:disableMandatoryCustomFieldValidation>true</platformMsgs:disableMandatoryCusto
mFieldValidation>
   <platformMsgs:disableSystemNotesForCustomFields>true   </platformMsgs:disableSystemNotesForC
ustomFields>
</platformMsgs:preferences>
</soap:Header>
```

Resetting Default Behavior

Important: The preferences described in this topic are only available to endpoints **2009.2** and lower. Note that these older endpoints are no longer supported. For details, see Support Has Ended for SuiteTalk 2009.2 and Earlier Endpoints.

Within the NetSuite UI, there are three types of default behaviors that may be associated with any particular record.

- Record fields can be automatically populated with default values.

- Records can have related fields that are automatically populated with default values when an initial value is entered. These fields are populated depending on the *condition* of the initial field.
- Records can be populated with a *calculated* value depending on the values set in a particular field.

When using web services, you may want to change the default behavior assigned to records since there is no visual confirmation of the default values being submitted. To specify your web services behavior, select one of the following options.

- **Use Conditional Defaults on Add** – Similar to the UI, if enabled, related fields are automatically populated with default values when a related value is entered in another field when *creating* a new record. If not enabled, no default values for conditional fields are submitted.
- **Use Conditional Defaults on Update** – If enabled, related fields can be automatically populated with default values when a value is entered while *updating* an existing record. If not enabled, no default values for conditional fields are submitted. This prevents overriding existing values that the user may not want to change.

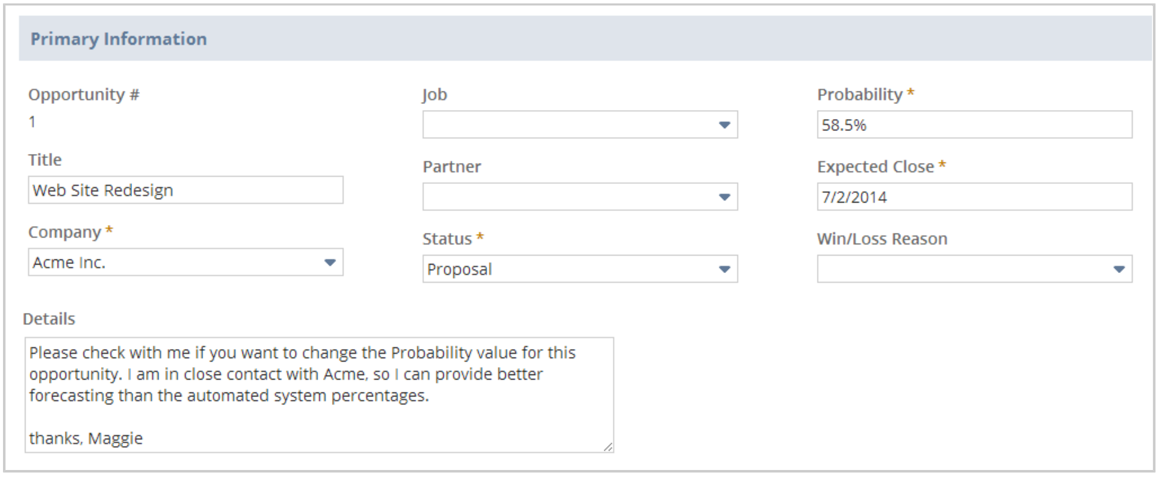
Both of these preferences are available at the company-wide and request level. For details, see Company-Wide Preferences and Request-Level Preferences.

Be aware that you **cannot** change the *default* behavior of calculated fields. Calculated fields are always reset when related fields are changed. However, you can override the value of the calculated field by submitting a value for that field in the request. Also, some fields within NetSuite are set as having slaving performed regardless of any default settings — the slaving values are mandatory. For these fields, unless a value is explicitly set, the field value is set as defined in the slaving definition regardless of default settings.

Example

When updating an Opportunity transaction, a change to the Status field causes the Probability field to automatically default to a new *conditional default* value. However, the Probability field can also be overridden by the user.

For example, when you set the status to Proposal, the Probability value automatically changes to 50.0%. However, a user could manually change that to some other number, as shown in the following screenshot.



Primary Information

Opportunity #: 1
Title: Web Site Redesign
Company *: Acme Inc.
Job:
Partner:
Status *: Proposal
Probability *: 58.5%
Expected Close *: 7/2/2014
Win/Loss Reason:

Details:
Please check with me if you want to change the Probability value for this opportunity. I am in close contact with Acme, so I can provide better forecasting than the automated system percentages.

thanks, Maggie

Therefore, it may be undesirable to have the field automatically populated with a conditional default value as part of a web services update operation. In that case, you would choose a value of false for the Use Conditional Defaults on Update preference.

Setting the Internal ID Preference

You can configure NetSuite to display internal ID values in the UI. This behavior can be useful during development of your web services integration. The display of internal IDs lets you verify that the values submitted in web services requests match the intended records.

To display internal ID values:

Note: The Show Internal IDs field is only available when at least one of the following features is enabled in your account: Client SuiteScript, Server SuiteScript, SuiteScript Server Pages, SuiteFlow, or Web Services (on the SuiteCloud tab) or Advanced Site Customization or SuiteCommerce Advanced (on the Web Presence tab). For more information, see the help topic Enabling Features.

1. Go to Home > Set Preferences. The **General** subtab is displayed by default.
2. In the **Defaults** section, click **Show Internal IDs**.

When this preference is enabled, most list views include Internal ID as one of the columns. For example, when you go to List > Relationships > Customers, the second column is Internal ID.

Changes to this preference affect only the current user.

Note: Another way to quickly determine the internal ID value for a record is through its URL. For example, when viewing a specific record, its URL may be something like the following: https://webservices.netsuite.com/app/common/entity/custjob.nl?id=272. In this example, the internal ID is 272.

Caching Behavior in Web Services

Preferences, customizations, and custom fields are cached by web services sessions, both in CRUD and search operations. For example, if you enable or disable an accounting preference at Setup > Accounting > Preferences > Accounting Preferences (Administrator), your current web services session still retains the previous setting of the preference.

To make sure that your web services requests respect your updated preferences, customizations, and changes in custom fields, you can use the following methods:

- Use the login operation to create a new web services session before sending a new web services request.
- If this is not possible, wait 60 minutes for the session timeout. After the timeout, the new web services session will respects the changes you made.

Integration Management

As of Version 2015 Release 2, all external applications that send web services requests to your NetSuite account are represented by one or more integration records. You can use integration records to manage applications, view activity logs, and more.

An integration record can be used to represent any application that sends web services requests or calls RESTlets. However, some capabilities of the integration record are limited in regard to RESTlet activity. This chapter focuses on using the integration record in conjunction with web services activity.

For more details on managing web services activity by using integration records, see the following topics:

- Integration Record Overview
- Adding an Integration Record
- Application Details for Client Code
- Sending an Application ID with User Credentials
- Token-Based Authentication Details
- Blocking Web Services Requests
- Web Services Execution Log
- Distributing Integration Records
- Default Web Services Integrations Record
- Using Integration Records in Sandbox Accounts
- Using Integration Records in Conjunction With SSO Calls
- Removing Integration Records
- Tracking Changes to Integration Records

Integration Record Overview

The integration record enhances your ability to manage and monitor web services requests sent to your NetSuite account. This capability is particularly useful if you have more than one application that sends requests. For example, your sales department may use one application to send data about new customers. Your HR department may use a different application to update employee records. Each application can be represented by a separate integration record.

For more details, see the following sections:

- Benefits of the Integration Record
- Application IDs
- Endpoint Requirements
- Ownership of Integration Records

Benefits of the Integration Record

When you use integration records to represent all of your external applications, you can do any of the following:

- **View details about each application** – Each integration record shows details such as the application's name and description.
- **Block an application** – You can use an integration record to block web services requests that come from the corresponding application. However, if you are using the 2015.1 or an earlier endpoint, be aware that some requests may be handled by the Default Web Services Integrations record, which has its own configuration options. For details, see Blocking Web Services Requests.
- **Select permitted authentication methods for an application** – When you create an integration record, you can permit the application to authenticate through user credentials, token-based authentication, or both. Note that when a request authenticates through user credentials, the request must also include an application ID, if you want the application's requests to be tracked by the appropriate integration record. (For details, see Application IDs.) Additionally, be aware that the permitted authentication methods vary depending on which endpoint the request uses. For details, see Endpoint Requirements.
- **Distribute an integration record** – You can bundle an integration record and make it available for installation in other NetSuite accounts. You can also configure a record so that it can be automatically installed in other accounts. For details, see Distributing Integration Records.
- **View an execution log specific to each application** – When you create and use an integration record for each application, that record's Web Services Execution Log lets you view only those requests sent by that application. If you are not using unique integration records for each application, you can view all requests together by using the log at Setup > Integration > Web Services Usage Log.
- **View a list of all your applications** – You can view a list of all your applications at Setup > Integration > Manage Integrations.

Application IDs

Every integration record has a unique application ID. This ID, a 32-character universally unique identifier, is generated when you create the record. The ID is displayed on the integration record in the Application ID field. However, application ID values are displayed only in the NetSuite account where they were created. So, if an integration record was created by a partner and distributed, that record's application ID is not visible to the partner's customers when they view the installed record in their own accounts.

In some cases, your web services requests are required to **include** an application ID. However, if your request uses token-based authentication, you must **omit** an application ID.

Use the following guidelines:

- Requests that authenticate through user credentials are required to include an application ID, if the request uses the 2015.2 endpoint or later. (For details, see Endpoint Requirements.) If the request uses the 2015.1 endpoint or earlier, an application ID is optional.
- Inbound SSO requests must include an application ID, if the requests use the 2015.2 endpoint or later. If the request uses the 2015.1 endpoint or earlier, an application ID is optional.
- Application IDs should not be included in outbound SSO calls, regardless of which endpoint is being used.
- If a request uses token-based authentication, you must not include an application ID, regardless of which endpoint you are using. If you include an application ID with a request that uses TBA, the request fails.

Although application IDs are considered identifiers and not authentication credentials, application IDs should nevertheless be considered confidential.

Important: Some customers may have application IDs that were provided to them by NetSuite Support prior to Version 2015 Release 2. These older IDs must not be used with the 2015.2 or a later WSDL. Older application IDs are permitted **only** in requests that use the 2015.1 WSDL or earlier. These requests are logged as part of the Default Web Services Integrations record's Web Services Execution Log.

If older application IDs are included in requests that use the 2015.2 or a later WSDL, the requests are denied.

You can identify an old application ID by its length. Old application IDs have a maximum of nine digits. By contrast, new applications IDs are 32-character UUIDs.

Endpoint Requirements

You can use integration records in conjunction with any supported endpoint. However, different requirements exist, depending on your endpoint.

2015.1 and Earlier

If you use the 2015.1 endpoint or earlier, be aware of the following:

- You have the option of letting some or all of your requests be tracked by a single integration record called Default Web Services Integrations. This record requires little maintenance, if any. The Default record was automatically created when your account was upgraded to Version 2015 Release 2 or subsequently, when you enabled the Web Services feature. The behavior of this record is slightly different from the behavior of integration records that you create. For details on this record, see Default Web Services Integrations Record.
- Token-based authentication is not supported. If you want a request to be tracked by a specific integration record (other than the Default record), it must authenticate through user credentials and include an application ID. For more details about application IDs, see Application IDs.

2015.2 and Later

If you upgrade to the 2015.2 or a later endpoint, all web services requests must be associated with an integration record other than the default record. That is, each request must include, either explicitly or implicitly, one of the following types of information:

- The application ID that was generated when the integration record was created, in conjunction with valid user credentials. For more details about application IDs, see Application IDs. If you use the login or ssoLogin operation, the application ID can be sent one time per session. For details, see Using Application ID with the Login and ssoLogin Operations.
- Token-based authentication details. For details on updating your code to authenticate this way, see Token-Based Authentication Details.

Note: Only one web services operation does not require authentication, nor any information identifying the application. It is the getDataCenterUrls.

2016.2 and Later

NetSuite 2016.2 includes two new permissions that make it possible for users other than administrators to view and manage integration records.

The List type Integration Applications permission provides view access to the Integrations list page.

The Full level of the Setup type Integration Application permission provides the ability to view, edit, and create integration records. This permission also provides access to the **Integration** field on the Web Services Operations search.

For details about web services searches, Searching for Web Services Log 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 editible by administrators in your account.

You can also install records in your account that were created elsewhere, as follows:

- Installed records can be installed as part of a bundle.
- In some cases, integration records can also be auto-installed. With this approach, a record is automatically installed when a request is sent that references the record's application ID. However, be aware that auto-installation is not recommended for records that are configured to permit token-based authentication. NetSuite recommends that these records be distributed through bundling.

Records that are installed through either of these methods are not fully editable, because they are considered to be 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 on these records, including the permitted authentication options and the Description field, can be changed only by an authorized user in the account that owns the records. When the owner makes changes, they are pushed automatically to your account. These changes are not reflected in the system notes that appear in your account.

For more information about creating integration records, see Creating an Integration Record. For more details about distributing integration records, see Distributing Integration Records.

Adding an Integration Record

For details about adding an integration record to a NetSuite account, see the following topics:

- Creating an Integration Record
- Auto-Installation of Integration Records
- Bundle Installation of Integration Records

Creating an Integration Record

You can add an integration record to your account by auto-installing the record, or by installing a bundle that includes the record. However, sometimes you must create an integration record by filling out a form in the UI. For example, if you have created an integration for your own use only, you should create an integration record by filling out the Integration form at Setup > Integration > Manage Integrations > New.

If you are a partner, you may want to provide customers with an integration record that represents an application that you have distributed. In these cases, you would create an integration record by using the procedure described in this topic. Later, you would distribute the integration record to customers, either through bundling or auto-installation. Whichever distribution method you choose, note that most of the integration record's fields will be read-only in the customer's account. The exceptions are the State field and the Note field. For help distribution an integration record, see Distributing Integration Records.

When you create an integration record, the system automatically generates an application ID. Additionally, depending on configuration choices you make, the system may automatically generate a consumer key and secret for the record.

Note: Only administrators and users with the Full Access role can complete this procedure.

To manually create an integration record:

1. Go to Setup > Integration > Manage Integrations > New.
2. In the **Name** field, enter a name for the integration. If you are a partner and you intend to distribute this record, note that this field will be read-only in the customer's account.
3. If appropriate, enter a description in the **Description** field. If you are a partner and you intend to distribute this record, note that this field will be read-only in the customer's account.
4. If you want NetSuite to block requests from this application, set the **State** dropdown list to **Blocked**. Otherwise, leave this field set to its default of **Enabled**. If you are a partner and you intend to distribute this record, be aware that the value you choose for this field is not propagated to customer accounts. The value of this field is always specific to one NetSuite account. For this reason, if the record is installed in another account, its value can vary. For example, if the record is installed by bundling, the State field is always initially set to Enabled. If the record is installed by auto-installation, its value depends on a preference. For details, see Auto-Installation of Integration Records.
5. If appropriate, enter additional details about the application in the **Note** field. If you are a partner and you intend to distribute this record, be aware that the text you enter in this field is not visible to customers. The value of this field is always specific to one NetSuite account.
6. On the **Authentication** subtab, select the authentication methods that this application should be permitted to use. You can choose one or both of the following:
   - User credentials – Lets the application authenticate by using the credentials associated with a single user: an email address, a password, a role, and a NetSuite account ID. For help updating an application to send a request that includes user credentials, see Sending an Application ID with User Credentials. This option also lets the application send inbound SSO requests. For details, see Using Integration Records in Conjunction With SSO Calls.
   - Token-based authentication – Lets the application use token-based authentication (TBA). Note that this method is supported only for requests that use the 2015.2 or later WSDLs. For help updating an application to send a request that includes token-based authentication details, see Token-Based Authentication Details.
7. Click **Save**.

   The system saves the new record and updates the page to show the record's application ID.

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

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

By contrast, application IDs are not considered authentication information but should still be considered confidential. For details, see Application IDs.

Auto-Installation of Integration Records

Auto-installation is the automatic addition of an existing integration record to your NetSuite account through the sending of a web services request. Users and administrators in your organization can use this method to add partners' integration records to your account.

Auto-installation is intended for use with integration records that are configured to authenticate through user credentials. By contrast, applications that permit token-based authentication should be distributed and installed through bundling, as described in Bundle Installation of Integration Records.

For more details about auto-installation, see the following sections:

- Understanding Auto-Installation
- Configuring the Require-Approval Preference

Note: When a record is installed through auto-installation (or through bundling), the values of many fields on the record are determined by the NetSuite account that owns the record. These fields are read-only in the account where the record is installed. For details, see Ownership of Integration Records.

Understanding Auto-Installation

Auto-installation occurs when a user sends a web services request that does both of the following:

- Authenticates by using user credentials. That is, the request must include valid Passport details: a user's email address, pasword, role, and NetSuite account ID. For more details, see Sending an Application ID with User Credentials.
- Includes the application ID of an integration record that is not already present in your NetSuite account.

When such a request is sent, the record is automatically installed.

Be aware that auto-installation can be triggered by any user with access to your NetSuite account, not exclusively account administrators.

The most common use of auto-installation occurs when users download and run applications provided by partners. For example, a partner may have begun distributing a newly created application that includes the application ID of a new integration record. Alternatively, a partner may have upgraded an existing application to use the 2015.2 or a later WSDL and include a new application ID.

It is also possible to auto-install an integration record from your sandbox to your production account. Or, if an integration record was removed from a sandbox account during a sandbox refresh, you can restore it by auto-installing it.

In all of these cases, after a request is sent that includes the application ID, the integration record is automatically installed. However, web services requests from the application may or may not be automatically permitted. This behavior varies depending on the configuration of a preference, as described in Configuring the Require-Approval Preference.

Configuring the Require-Approval Preference

The configuration of the **Require Approval during Auto-Installation of Integration** preference determines whether automatically installed integration records are enabled at the time they are installed.

This preference can be configured in either of the following ways:

- False (default) – The State field on each newly installed record is automatically set to Enabled, and all requests are permitted.
- True – The State field is set to Waiting for Approval. In this case, all requests sent by that application are blocked until you edit the record and set the State field to Enabled.

In other words, if you as an administrator want to review and manually approve each newly installed integration, you should set this preference to true. By contrast, if you are confident that all newly installed applications should be granted access, set this preference to false.

You can manage this preference at Setup > Integration > Web Services Preferences.

For more details about using the State field, see Blocking Web Services Requests.

Bundle Installation of Integration Records

One way to install an integration record is to install a bundle that includes that record. You can use this method to add partners' integration records to your account, or to install a record created in a sandbox into a production account.

Be aware that, when you use this approach, the State field on the new record is initially set to Enabled. This configuration means that web services requests from the application are permitted. (By contrast, when you install an integration record by auto-installation, the value of the State field is determined by how you have configured the **Require Approval during Auto-Installation of Integration** preference.)

After the record is installed, you can edit the record and update the State field as needed. For more details, see Blocking Web Services Requests.

For general help installing a bundle, see the help topic SuiteApp Installation and Update.

Note: When a record is installed through auto-installation (or through bundling), the values of many fields on the record are determined by the NetSuite account that owns the record. These fields are read-only in the account where the record is installed. For details, see Ownership of Integration Records.

Application Details for Client Code

When you are using the integration record to track an application, each request must include data that identifies the application. For that reason, after you create or install an integration record, you may have to update the client code.

There are two ways to identify the application: by using the application ID, or by using token-based authentication (TBA). Note that it is not permitted to send both types of data. For details, see the following topics:

- Sending an Application ID with User Credentials

- Token-Based Authentication Details

Sending an Application ID with User Credentials

When you track an application by using the integration record, the application can authenticate in one of two ways: by sending an application ID and user credentials, or by using token-based authentication. For help with token-based authentication, see Token-Based Authentication Details. For details about sending an application ID, see the following topics:

- Requirements for Using an Application ID and User Credentials
- Application ID Code Samples (Request-Level Credentials)
- Using Application ID with the Login and ssoLogin Operations

Requirements for Using an Application ID and User Credentials

If you want your request to authenticate by using user credentials, and you want to track the application by using a specific integration record, the following must be true:

- An integration record for the application must exist at Setup > Integration > Manage Integrations. On the integration record, the User Credentials option must be enabled. For help creating an integration record, see Creating an Integration Record.
- Your request must include, either explicitly or implicitly, the application ID that was generated when the integration record was created. If the integration was created in your NetSuite account, you can view the application ID on the integration record. If the integration record was created by a partner, in most cases you will obtain an updated integration from the partner that includes the application ID. With request-level credentials, the application ID must be included in the header of each request. For the login or ssoLogin request, you have an additional option, as described in Using Application ID with the Login and ssoLogin Operations.

  The request must use a Passport object, not a TokenPassport. For examples of how to include Passport data in a request, see Authentication for Web Services.

Note that you can use either request-level credentials or the login operation. You can use any supported WSDL. NetSuite supports WSDLs for three years. For more details about WSDL support, see Support for Existing WSDL Versions.

Application ID Code Samples (Request-Level Credentials)

The following procedure shows one way of updating an application to include an application ID, which is a field in the ApplicationInfo complex type. In the following C# code snippets, application ID is added to an integration's request-level credentials.

For Java code samples, download the Java sample applications from the SuiteTalk Sample Applications site.

The ApplicationInfo complex type is defined in the Messages XSD.

To add an application ID to an integration:

1. Create an ApplicationInfo class member. For example:

```
namespace NetSuite
{
```

```
class SuiteTalkApplication
{

      public NetSuiteService _service;
      private Preferences _prefs;
      private SearchPreferences _searchPreferences;
      private ApplicationInfo _appInfo;
```

2. Create an instance of ApplicationInfo. Set the applicationId value to the actual application ID of the relevant integration record.

```
private void setApplicationInfo()
{

      _appInfo = new ApplicationInfo();
      _service.applicationInfo = _appInfo;
      _appInfo.applicationId = "C4A2541D-16FB-4845-B8A2-8B8A34BDBB61";

}
```

3. Use your instance of ApplicationInfo to modify the SOAP header that will be generated.

```
public SuiteTalkCourse()
{
   ServicePointManager.ServerCertificateValidationCallback += delegate { return true; };

   _service = new DataCenterAwareNetSuiteService();
   setPreferences();
   setPassport();
   setApplicationInfo();

}
```

The next time the application sends a request, the application ID is included in the header.

```
<soap:Header>
   <applicationInfo xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <applicationId>A7EE88F2-9DAA-43A2-9940-29AAA0E2DA61</applicationId>
   </applicationInfo>
   <passport xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <email xmlns="urn:core_2017_1.platform.webservices.netsuite.com">user@wsuser.com</email>
      <password xmlns="urn:core_2017_1.platform.webservices.netsuite.com">**********</password>

      <account xmlns="urn:core_2017_1.platform.webservices.netsuite.com">1013519</account>
      <role internalId="3" xmlns="urn:core_2017_1.platform.webservices.netsuite.com"/>
   </passport>
   <preferences xmlns="urn:messages_2017_1.platform.webservices.netsuite.com"/>
</soap:Header>
```

Using Application ID with the Login and ssoLogin Operations

If you use the login or ssoLogin operation, note the following: You can define an application ID in the SOAP header of a login or ssoLogin request. In this case, all requests in the session will be identified with that ID. Alternatively, you can define the application ID within the header of each request in the session. If you choose the latter method, be aware that all requests within a single session must use the same application ID. If any request uses a different application ID, the system returns an error.

Token-Based Authentication Details

For details about using token-based authentication (TBA) with web services, see the following topics:

- Requirements for Using Token-Based Authentication
- Regenerating a Consumer Key and Secret
- TokenPassport Complex Type
- Updating an Integration to Send Token-Based Authentication Details
- Web Services Governance for Token-Based Authentication
- Token-Based Authentication Errors in Web Services

Requirements for Using Token-Based Authentication

To use token-based authentication (TBA) during a web services request, you must meet several requirements, as described in the following sections:

- General Setup Requirements
- Additional Web Services Requirements

General Setup Requirements

Before you can use token-based authentication, you must complete several setup tasks. These tasks include the following:

- The Token-based Authentication feature must be enabled. For details, see the help topic Enabling the Token-based Authentication Feature.
- You must have created a role that permits login by using token-based authentication. For details, see the help topic Setting Up Token-based Authentication Roles.
- You must have assigned a user to a role that has permission to log in by using token-based authentication. For details, see the help topic Assigning Users to Token-based Authentication Roles.
- An integration record for the application must exist at Setup > Integration > Manage Integrations. On the integration record, the Token-based Authentication option must be enabled. For help creating an integration record, see Creating an Integration Record.
- You must have the consumer key and secret that were generated when the relevant integration record's Token-based Authentication option was enabled. If you do not have these credentials, you can generate new ones, for applications created in your NetSuite account. Note that if the record was distributed via bundling, the credentials were included in the bundle, and only the owner of the application can regenerate them. For details, see Regenerating a Consumer Key and Secret.
- You must have created a token and token secret for the user who will send the web services request for this application. For details on this process, see the help topic Managing TBA Tokens.

Additional Web Services Requirements

For an application to use token-based authentication during a web services request, all of the following must be true:

- The request must be made by using the 2015.2 WSDL or later.
- The request must use request-level credentials, not the login operation. For details on the difference, see Authentication for Web Services.

- The request must use a TokenPassport object, not a Passport object. For help updating an integration to use a TokenPassport, see TokenPassport Complex Type and Updating an Integration to Send Token-Based Authentication Details.

Regenerating a Consumer Key and Secret

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

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

If the record was installed in your account through bundling, you cannot reset the credentials. The credentials can be reset only by an authorized user in the NetSuite account where the record was created.

Note: Only administrators and users with the Full Access role can complete this procedure.

To regenerate a consumer key and secret:

1. Go to Setup > Integration > Manage Integrations.
2. Select the record for which you want to generate a new consumer key and secret.

   The record opens in view mode.
3. Click the **Edit** button.
4. Click the **Reset Credentials** button.

   The system displays a popup message asking if you are sure you want to reset the credentials.
5. Click **OK**.

   The system resets the credentials. The record is again shown in view mode, with the new consumer key and secret displayed.

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

TokenPassport Complex Type

A web services request that uses token-based authentication must use the TokenPassport complex type. By contrast, a request that authenticates in another way uses either the Passport or SsoPassport type.

The TokenPassport references the TokenPassportSignature complex type, which is another important element in the token-based authentication process. Both complex types are defined in the Core XSD file.

Important: Do not use TokenPassport and Passport SOAP headers in a single request. Mixing different authentication types in a single SOAP message returns a SOAP fault.

For more details about these complex types, see the following sections:

- TokenPassport
- TokenPassportSignature

TokenPassport

The TokenPassport complex type uses the following fields. All are required:

Field	Description
account	Your NetSuite account ID. You can find this number at Setup > Integration > Web Services Preferences, in the Account ID field.
consumerKey	The consumer key for the integration record. This string was created when you checked the Token-based Authentication box on the integration record and saved it. This string is shown only one time, when it is first generated. If you no longer have this string, you can reset the credentials, as described in Regenerating a Consumer Key and Secret.
token	This is a string identifier or an ID of a token that represents a unique combination of a user, a role, and an integration record. This string can be generated in multiple ways. For details, see the help topic Managing TBA Tokens.
nonce	This field should hold a unique, randomly generated alphanumeric string of 6–64 characters.
timestamp	This field should hold a current timestamp in Unix format.
signature	The signature is a hashed value. You create this value by using all of the other values in the TokenPassport, plus the appropriate token secret and consumer secret. Along with the actual signature, you must identify the algorithm used to create the signature. For details, see TokenPassportSignature.

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

TokenPassportSignature

You use the TokenPassportSignature complex type to identify the signature, which is a hashed value. The TokenPassportSignature also includes an attribute labeled algorithm, which you use to identify the algorithm used to create the signature.

At a high level, you create the signature by completing the following steps:

1. Create a **base string**. The base string is variable created from concatenating a series of values specific to the request. Use an ampersand as a delimiter between values. The values should be arranged in the following sequence:
   - NetSuite account ID
   - Consumer key
   - Token
   - Nonce (a unique, randomly generated alphanumeric string, with a minimum of six characters and maximum of 64)
   - Timestamp

   For example, suppose you have the following variables:
   - **NetSuite account ID** – 1234567

- **Consumer key** – 71cc02b731f05895561ef0862d71553a3ac99498a947c3b7beaf4a1e4a29f7c4
- **Token** – 89e08d9767c5ac85b374415725567d05b54ecf0960ad2470894a52f741020d82
- **Nonce** – 6obMKq0tmY8ylVOdEkA1
- **Timestamp** – 1439829974

In this case, the base string would be as follows.

```
1234567&71cc02b731f05895561ef0862d71553a3ac99498a947c3b7beaf4a1e4a29f7c4&89e08d9767c5ac85b37441
5725567d05b54ecf0960ad2470894a52f741020d82&6obMKq0tmY8ylVOdEkA1&1439829974
```

2. Create a **key**. The key is a string variable created by concatenating the appropriate consumer secret and token secret. These two strings should be conctatenated by using an ampersand.

   For example, suppose you have the following variables:

   - **Consumer secret** – 7278da58caf07f5c336301a601203d10a58e948efa280f0618e25fcee1ef2abd
   - **Token secret** – 060cd9ab3ffbbe1e3d3918e90165ffd37ab12acc76b4691046e2d29c7d7674c2

   In this case, the key would be as follows:

```
7278da58caf07f5c336301a601203d10a58e948efa280f0618e25fcee1ef2abd&060cd9ab3ffbbe1e3d3918e90165ff
d37ab12acc76b4691046e2d29c7d7674c2
```

3. Choose a **supported hash algorithm** to create an RFC 2104-compliant signature. Supported algorithms include HMAC-SHA1 and HMAC-SHA256.
4. Use the base string, the key, and the algorithm to create an RFC 2104-compliant signature. The signature must be encoded by Base64.

Note: For an example of how to represent the signature and algorithm in a SOAP request, see Token-Based Authentication Details.

Updating an Integration to Send Token-Based Authentication Details

The following procedure and Java code snippets show one way of updating an application to include token-based authentication (TBA) details.

To send token-based authentication details, you use the TokenPassport and TokenPassportSignature complex types. These types are defined in the Core XSD. Refer also to TokenPassport Complex Type.

Note that this example uses request-level credentials. When using token-based authentication, your SOAP request must be structured this way. The login operation is not supported. For additional requirements, see Requirements for Using Token-Based Authentication.

Note: If a request uses token-based authentication, your request must not include an application ID. If you include an application ID with a request that uses TBA, the request fails.

To update an integration to send token-based authentication details:

1. Add logic for creating an RFC 2104-compliant signature. For example, you could add a method that has three arguments: a base string, a key, and an algorithm. Later, you could call this

method by passing values specific to your request. (For details on the data needed to create the correct base string and key values, see TokenPassportSignature.)

This example does not require the use of a third-party library. However, the javax.crypto package is used.

```
protected String computeShaHash(String baseString, String key, String algorithm) throws Excepti
on
{
   if (!algorithm.equals("HmacSHA256") && !algorithm.equals("HmacSHA1")) algorithm = "HmacSHA1"
;

   {
      byte[] bytes = key.getBytes();
      SecretKeySpec mySigningKey = new SecretKeySpec(bytes, algorithm);

      Mac messageAuthenticationCode = Mac.getInstance(algorithm);

      messageAuthenticationCode.init(mySigningKey);

      byte[] hash = messageAuthenticationCode.doFinal(baseString.getBytes());

      String result = new String(Base64.encodeBase64(hash, false));

      return result;
   }
}
```

2. Using variables that reflect data specific to your request, add logic for calling the computeShaHash method you created in step 1 to calculate a signature.

```
   public String computeSignature(
         String account,
         String consumerKey,
         String consumerSecret,
         String token,
         String tokenSecret,
         String nonce,
         Long timeStamp,
         String signatureAlgorithm
   ) throws Exception
   {
      String baseString = account + "&" + consumerKey + "&" + token + "&" + nonce + "&" + timeS
tamp;

      String key = consumerSecret + '&' + tokenSecret;

      String signature;

      if (signatureAlgorithm.equals(SignatureAlgorithm._HMAC_SHA256))
      {
         signature = computeShaHash(baseString, key, "HmacSHA256");
      } else {
         signature = computeShaHash(baseString, key, "HmacSHA1");
      }
      return signature;
   }
```

3. Add logic for creating a TokenPassport SOAP header.

```
@Override
   public void setTokenPassport(
         String account,
         String consumerKey,
         String token,
         String nonce,
         Date timestamp,
         String signatureAlgorithm,
         String signature
   ) throws Exception
   {
      initPort();
      if(m_tokenPassport == null)
      {
         m_tokenPassport = new TokenPassport();
         ((Stub) m_port).setHeader( m_sMessagesUrn, "tokenPassport", m_tokenPassport);
      }
      m_tokenPassport.setAccount(account);
      m_tokenPassport.setConsumerKey(consumerKey);
      m_tokenPassport.setToken(token);
      m_tokenPassport.setNonce(nonce);
      m_tokenPassport.setTimestamp(timestamp.getTime());
      TokenPassportSignature signatureElement = new TokenPassportSignature();
      signatureElement.setAlgorithm(signatureAlgorithm);
      signatureElement.set_value(signature);
      m_tokenPassport.setSignature(signatureElement);
   }
```

4. To actually add your token-based authentication credentials to a request, add logic similar to the following. Notice that in this example, the exact variables are defined. This example calls the computeSignature method, from step 3, to create the signature. It also sets the values in the fields of the TokenPassport object created in step 4. These values are later visible in the SOAP request. By contrast, the consumer secret and token secret are not explicitly part of the SOAP request. They are represented only in the signature.

```
public void testGetServerTimeWithTBA() throws Exception
{
// Constants
   String sCompId = "1234567";
   String consumerKey = "71cc02b731f05895561ef0862d71553a3ac99498a947c3b7beaf4a1e4a29f7c4";
   String consumerSecret = "7278da58caf07f5c336301a601203d10a58e948efa280f0618e25fcee1ef2abd";
   String tokenId = "89e08d9767c5ac85b374415725567d05b54ecf0960ad2470894a52f741020d82";
   String tokenSecret = "060cd9ab3ffbbe1e3d3918e90165ffd37ab12acc76b4691046e2d29c7d7674c2";
   String algorithm = SignatureAlgorithm._HMAC_SHA256;

// Nonce and timeStamp must be unique for each request. Also, make sure that the nonce value yo
u
// generate does not contain special characters.

   String nonce = RandomStringUtils.randomAlphanumeric(20);
   Long timeStamp = System.currentTimeMillis() / 1000L;

   String signature = c.computeSignature(sCompId, consumerKey, consumerSecret, tokenId, tokenSe
cret, nonce, timeStamp, algorithm);

// create and set TokenPassport SOAP Header
// notice - no secrets are included in this call
      c.setTokenPassport(
            sCompId,
            consumerKey,
            tokenId,
            nonce,
            new Date(timeStamp),
            algorithm,
            signature);

      c.getPort().getServerTime();
   }
```

The following example shows how the resulting SOAP request should appear.

```
<ns:tokenPassport soap:actor="http://schemas.xmlsoap.org/soap/actor/next" soap:mustUnderstand="
0" xmlns:ns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <ns:account>1234567</ns:account>
   <ns:consumerKey>71cc02b731f05895561ef0862d71553a3ac99498a947c3b7beaf4a1e4a29f7c4</ns:consume
rKey>
   <ns:token>89e08d9767c5ac85b374415725567d05b54ecf0960ad2470894a52f741020d82</ns:token>
   <ns:nonce>6obMKq0tmY8ylVOdEkA1</ns:nonce>
   <ns:timestamp>1439829974</ns:timestamp>
   <ns:signature algorithm="HMAC_SHA256">fzGxUBu6SZvGqv5hk8P4ou2DPthSxXtJ4zJIeCBQK5A=</ns:signa
ture>
</ns:tokenPassport>
```

Note: For the signature algorithm value, you can use either a hyphen or an underscore. (For example, HMAC-SHA256 or HMAC_SHA256) However, the underscore is preferred. This format is used by the SignatureAlgorithm enumeration, which is defined in the core types XSD.

Web Services Governance for Token-Based Authentication

When requests are sent by using token-based authentication, SuiteTalk uses the following governance guidelines according to the earlier concurrency governance model. If you have a SuiteCloud Plus license and have been designated a concurrent web services user, you are permitted 10 concurrent requests. Otherwise, you are permitted one request at a time.

If you exceed these limits, the system returns the ExceededConcurrentRequestLimitFault. The corresponding error code is `WS_REQUEST_BLOCKED`.

According to the web services concurrency governance introduced in version 2017.2, a single user using token-based authentication can send parallel requests up to the account limit, and the earlier web services user concurrency limits do not apply.

Important: You can benefit from this option if the web services and REST concurrency governance is enabled in your account. To check whether the concurrency governance is enabled in your account, see the Web Services and RESTlet Account Governance Field on the Web Services Preferences page, at Setup > Integration > Integration Management > Web Services Preferences (Administrator).

Additionally, the account limit is lenient, and it is possible to temporarily exceed the limit.

For detailed information about the changes in web services concurrency governance, see Web Services and RESTlet Concurrency Governance Starting From Version 2017.2.

Token-Based Authentication Errors in Web Services

Check the following sections for information about the most common issues that can occur when you use token-based authentication with web services, and for information about troubleshooting these issues.

Ambiguous Authentication Errors

When you use token-based authentication (TBA) in web services, an ambiguous authentication error response is returned if you use another authentication mechanism together with the TBA header.

You receive this error if besides the TBA header, your request contains an Application ID, a passport object with an email address and a password, or a valid JSESSIONID.

The error occurs in the following cases:

- If a single web services request contains a combination of the Passport, TokenPassport and SsoPassport complex types.
- If a single web services request contains both the tokenPassport and the ApplicationInfo complex types, and therefore contains the application ID in the SOAP header.

Other Invalid Login Errors

Apart from ambiguous authentication errors, the following authentication errors may occur.

Issues with Nonce and Invalid Timestamps

The nonce and timestamp values must be unique for each request. Reusing these values is not allowed. If the nonce had already been used, or if the timestamp is different from the correct time, an error message is returned.

Additionally, you must make sure that the nonce value you generate does not contain special characters.

Issues with Sending Multiple TokenPassports in a Single Request

Sending multiple TokenPassports in the same request is not allowed and results in an invalid login error. The SOAP request must contain exactly one TokenPassport. A TokenPassport consists of a nonce, a timestamp, and a TokenPassportSignature. Each TokenPassport can only be used once.

For information about how to construct a TokenPassport, including code samples, see Updating an Integration to Send Token-Based Authentication Details.

Issues with Sending the Same TokenPassport in Multiple Requests

Sending the same TokenPassport in multiple requests is not allowed, and results in an invalid login error. You must make sure that you generate a new TokenPassport, complete with a new nonce value and the correct timestamp for every request. One TokenPassportSignature can only be used once, even if your code contains retry logic to send a SOAP request again if a request fails.

For information about how to correctly construct a TokenPassportSignature, including code samples, see Updating an Integration to Send Token-Based Authentication Details.

Troubleshooting Issues with Token-Based Authentication

See the following section for information about investigating authentication issues.

Logging SOAP Requests and Responses

In general, you should log the full content of all SOAP requests and responses while you develop your web services integration. This way you can troubleshoot your code, find out whether you use multiple authentication methods in the same request, and also find out if subsequent requests are sending the same nonce and timestamp.

Note: Logging the full content of SOAP requests and responses is also useful because if a request does not pass validation, it is not logged in the web services usage log. Additionally, even when a request using token-based authentication is logged, the authentication details are masked in the logs for security reason. This means that you cannot use the log for investigating authentication issues.

When a Passport and a TokenPassport are sent in the same request, the following SOAP response is returned:

```
      <soapenv:Body>
         <soapenv:Fault>
            <faultcode>soapenv:Server.userException</faultcode>
            <faultstring>Ambiguous authentication</faultstring>
            <detail>
               <ns1:hostname xmlns:ns1="http://xml.apache.org/axis/">f-partners-java001.svale.n
etledger.com</ns1:hostname>
            </detail>
         </soapenv:Fault>
      </soapenv:Body>
```

When a TokenPassport and the ApplicationInfo are sent in the same request, the following SOAP response is returned:

```
      <soapenv:Body>
         <soapenv:Fault>
            <faultcode>soapenv:Server.userException</faultcode>
            <faultstring>Ambiguous authentication</faultstring>
            <detail>
               <platformFaults:invalidCredentialsFault xmlns:platformFaults="urn:faults_2017_1.
platform.webservices.netsuite.com">
                  <platformFaults:code>USER_ERROR</platformFaults:code>
                  <platformFaults:message>Ambiguous authentication</platformFaults:message>
               </platformFaults:invalidCredentialsFault>
               <ns1:hostname xmlns:ns1="http://xml.apache.org/axis/">f-partners-java002.svale.n
```

```
etledger.com</ns1:hostname>
            </detail>
         </soapenv:Fault>
      </soapenv:Body>
```

Using the Login Audit Trail

The Login Audit Trail is a specialized search that helps keep track of account users, when they have logged in, and from where.

By using the Login Audit Trail search, you can also find out details about why a web services request failed to authenticate.

To define an advanced Login Audit Trail search to see details of web services login requests:

1. Go to Setup > Users/Roles > User Management > View Login Audit Trail (Administrator).
2. Check the **Use Advanced Search** box.
3. Click the Results subtab to define the columns that are displayed in the search results. To view more verbose information about login attempts, add the Detail result field.
4. Click Submit to run the search and display the results.

The Detail column in the search results contains more information about the login attempts, and provides information about the reason for the authentication failure.

The following messages indicate some of the most common issues:

- NonceUsed — The combination of nonce and timestamp was already used by the same user.
- InvalidTimestamp — The timestamp provided is different from the correct server time by more than five minutes.
- InvalidSignature — The request was not signed correctly.

For more information about defining Login Audit Trail searches, see the help topic Login Audit Trail Overview.

Blocking Web Services Requests

You can use an integration record to block web services requests that come from the corresponding application. However, when you use this method, you block only those requests that include data that identifies that integration record. For example, if you block Integration Record A, in essence you are blocking only those requests that include either Integration Record A's application ID or its token-based authentication details.

An application that is using the 2015.1 or an earlier endpoint is not required to send this data in requests. Requests that omit this data are handled by the Default Web Services Integrations record, which can also be blocked.

To block web services requests:

1. Do one or both of the following:
   - To block requests that are identified with a particular integration record, open that record for editing. Set the **State** field to **Blocked**, and click **Save**. After you take this action, your NetSuite account will not accept requests that use either the application ID or token-based authentication details specific to this record. Note also that the system will block any RESTlet calls made using the consumer key and secret associated with this record.

- To block web services requests from all external applications that omit an application ID or token-based authentication details, open the Default Web Services Integrations record for editing. Set the **State** field to **Blocked**, and click **Save**.

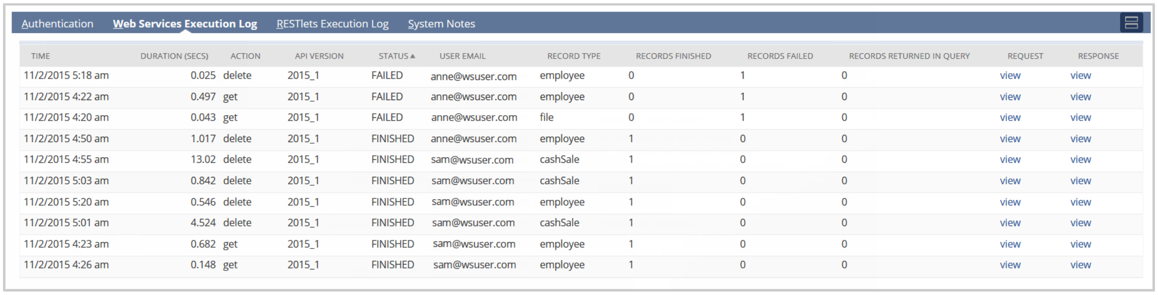
Web Services Execution Log

Each integration record includes a subtab labeled Web Services Execution Log. This log lists web services calls that are uniquely identified with the integration record you are currently viewing. That is, the log includes those requests that referenced one of the following:

- The integration record's application ID.
- The integration record's consumer key.

For each request, the log includes details such as the following:

- The date and time that the request was made.
- The duration of the request.
- Information about the requests that were rejected due to a violation of the concurrency governance.
- The WSDL version used (under the heading **API Version**).
- The email address of the user who sent the request.
- If applicable, the record type that was the subject of the request.
- Links to the SOAP request and response.



TIME	DURATION (SECS)	ACTION	API VERSION	STATUS ▲	USER EMAIL	RECORD TYPE	RECORDS FINISHED	RECORDS FAILED	RECORDS RETURNED IN QUERY	REQUEST	RESPONSE
11/2/2015 5:18 am	0.025	delete	2015_1	FAILED	anne@wsuser.com	employee	0	1	0	view	view
11/2/2015 4:22 am	0.497	get	2015_1	FAILED	anne@wsuser.com	employee	0	1	0	view	view
11/2/2015 4:20 am	0.043	get	2015_1	FAILED	anne@wsuser.com	file	0	1	0	view	view
11/2/2015 4:50 am	1.017	delete	2015_1	FINISHED	anne@wsuser.com	employee	1	0	0	view	view
11/2/2015 4:55 am	13.02	delete	2015_1	FINISHED	sam@wsuser.com	cashSale	1	0	0	view	view
11/2/2015 5:03 am	0.842	delete	2015_1	FINISHED	sam@wsuser.com	cashSale	1	0	0	view	view
11/2/2015 5:20 am	0.546	delete	2015_1	FINISHED	sam@wsuser.com	employee	1	0	0	view	view
11/2/2015 5:01 am	4.524	delete	2015_1	FINISHED	sam@wsuser.com	cashSale	1	0	0	view	view
11/2/2015 4:23 am	0.682	get	2015_1	FINISHED	sam@wsuser.com	employee	1	0	0	view	view
11/2/2015 4:26 am	0.148	get	2015_1	FINISHED	sam@wsuser.com	employee	1	0	0	view	view

The data shown is similar to the data shown in the Web Services Usage Log. However, one difference is that the Web Services Execution Log shows the version of the WSDL that was used. The Usage Log does not.

Another difference between the two logs is that the Web Services Usage Log is not specific to any one integration record. For more details about the Web Services Usage Log, see Using the Web Services Usage Log.

Note: In production environments, the data is accessible for 21 days. In sandbox environments, the data is accessible for seven days.

Distributing Integration Records

In some cases, it may be necessary to distribute an integration record to other NetSuite accounts. Similarly, if you created a record in your sandbox account, you might want to distribute it to your production account. You can also do the reverse: install a record created in your production account into your sandbox.

For all of these situations, you have the following options:

- Distributing by Auto-Installation
- Distributing by Bundling

Distributing by Auto-Installation

When certain conditions are met, a user can auto-install an integration record into a NetSuite account or account by sending a web services request to the target account.

If you want your integration record to be installed in this manner, note that the record must be configured to permit authentication through user credentials. If the only permitted authentication method is token-based authentication, the record cannot be installed by using this method.

One advantage of distributing by auto-installation is that users have some control over the value of the State field on newly installed records. The initial value of this field is set according to the configuration of the **Require Approval during Auto-Installation of Integration** preference.

To distribute an integration record by using auto-installation:

1. After the record has been created, open the record for editing and verify that the User Credentials box is checked. Additionally, make a note of the record’s application ID.
2. Update the code for the integration to include the application ID associated with the record.
3. Do one or both of the following, as needed:
   - Provide the updated integration or the application ID to the appropriate users. For example, if you are distributing the integration to your customers, you might want to provide them with the updated integration code and instruct them to send a request to the account or account where the record should be installed. You may also want to instruct users to check the configuration of the **Require Approval during Auto-Installation of Integration** preference. If this preference is set to true, the new integration record will be blocked until it is manually enabled.
   - Send a request to the account or account where you want to install the record.

   When the request is received, the record is automatically installed. If the State field is set to Waiting for Approval, an account administrator must manually enable the integration before requests can be accepted. That is, an administrator must open the record for editing and set the State field to Enabled.

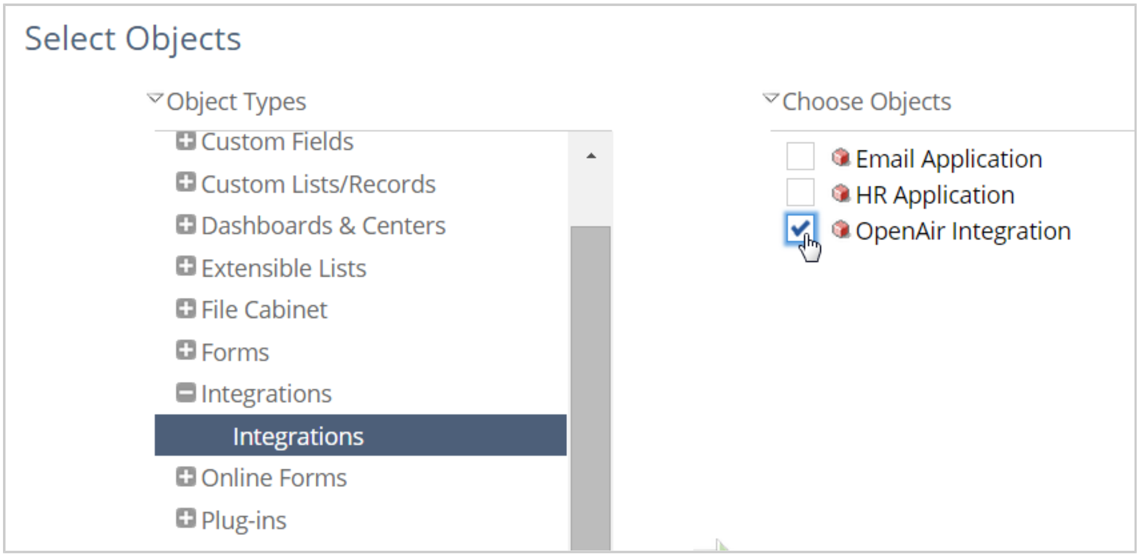
See also Auto-Installation of Integration Records.

Distributing by Bundling

The integration record is supported for bundling with SuiteBundler.

If the only permitted authentication method for the record is token-based authentication, you should use this method. This method may also be preferable if you have additional customizations that you want to distribute.

To include integration records in a bundle, select the appropriate records on the SuiteBundler’s Select Objects page. In the Object Types column, select Integrations > Integrations. Then select the appropriate records in the Choose Objects column. This column shows both active and inactive integration records.



See also Bundle Installation of Integration Records.

Note: Note that when you distribute an integration record by bundling, its State field is initially set to Enabled in the target account. By contrast, if you distribute the record by auto-installation, the initial value is determined by the configuration of the **Require Approval during Auto-Installation of Integration** preference.

Note: If the integration record is installed in your account by bundling, you cannot reset the credentials. The credentials can be reset only by an authorized user in the NetSuite account where the record was created. For more information, see Regenerating a Consumer Key and Secret.

Default Web Services Integrations Record

If you use web services, your account includes an automatically generated integration record called Default Web Services Integrations. This record represents every request sent to your account that does not include data that identifies a specific integration record. Put another way, these requests do not include either of the following:

- A 32-character application ID
- Token-based authentication details

Such requests are permitted **only** if the request uses the 2015.1 or a previous endpoint. If the request uses the 2015.2 endpoint or later, the request **must** include either a 32-character application ID or token-based authentication details. Otherwise, the request is denied.

In some cases, a request may include an old application ID that was created by using a previous version of NetSuite. This type of request is also handled by the Default Web Services Integrations record. These requests are not permitted with the 2015.2 WSDL or later.

The Default Web Services Integrations record has many of the same qualities as an integration record that was created to track one specific external application. For example, the Default integrations record includes a Web Services Execution Log that shows requests and responses. The Default record can also be blocked. However, you cannot enable token-based authentication for the Default integrations record.

Exercise caution when editing the Default Web Services Integrations record. Before you create integration records to represent your external applications, this record represents all of your web services traffic.

Using Integration Records in Sandbox Accounts

Important: This content applies only to sandbox accounts for customers that have NA sandboxes

Any integration record you create in a sandbox account can be installed in your production account, either through bundling or auto-installation. You can also install records in your sandbox that were created elsewhere, either in your production account or in another NetSuite account.

For details, see Auto-Installation of Integration Records and Bundle Installation of Integration Records.

If an integration record is created in your primary sandbox account and later installed in your production account, after a sandbox refresh, the integration record settings are copied from your production account to your primary sandbox account. For details, see Sandbox Refresh Impact on Integration Records.

Recommended development process for integration records:

1. Create an integration record in the primary sandbox account.
2. Test the integration record in the primary sandbox.
3. Transfer the integration record to the production account either by auto-installation or by bundle.

When using integration records in a sandbox account, be aware that there are a few behaviors that differ compared with integration records used only in production accounts. For more details, see the following topics:

- Fields Shared Between Production and Primary Sandbox Accounts
- Primary Sandboxes Versus Secondary Sandboxes
- Sandbox Refresh Impact on Integration Records

Note: For details on using Token-based Authentication with integration records with production and sandbox accounts, see Token-based Authentication Credentials and Accounts.

Fields Shared Between Production and Primary Sandbox Accounts

Integration records have fields that are specific for each production account, and fields that are shared between production and primary sandbox accounts. The following fields are shared between production and primary sandbox accounts:

- Name
- Application ID
- Consumer Key
- Consumer Secret

The owner, or creator of the integration record, can edit these shared fields in both production and sandbox accounts. The changes that are made to these fields in one account are automatically duplicated in the other account.

Note: For details on distributing an integration record, see Distributing Integration Records.

Primary Sandboxes Versus Secondary Sandboxes

In some cases, you may have to distribute an integration record from a sandbox to your production account, or you may need to do the reverse. In these cases, the behavior of the installed integration record varies depending on whether you are working with a primary or secondary sandbox account.

Your primary sandbox is the one that has the same NetSuite account ID as your production account. By contrast, a secondary sandbox has a NetSuite account ID with a suffix such as _SB2 or _SB3. You can view the account IDs of your sandbox accounts at Setup > Company > Sandbox Accounts.

If you create an integration record in your primary sandbox account and install it in your production account, the production record can be edited as if it had been manually created in the production account. The reverse is also true. (If you created an integration record in your production account and install it in your primary sandbox, it is fully editable in this account.) Changes made to the integration record in one of these accounts are automatically duplicated in the other account.

In contrast, if you create a record in a secondary sandbox and install it in your production account, the record in the production account is not fully editable. It behaves as if it came from a NetSuite account. For an integration record created in a secondary sandbox account, you can edit only the State and Note fields in the production account.

Sandbox Refresh Impact on Integration Records

When a sandbox account is refreshed, data that was created in the sandbox account and not in production is overwritten with production data. A sandbox refresh is one of the few ways that an integration record can be uninstalled from a NetSuite account. That is, if a record was created in your sandbox account but not installed in your production account, it is removed as part of the sandbox refresh. After the refresh, the integration record can be auto-installed back into the sandbox account, if the integration record was configured to permit authentication with user credentials, and if you have the application ID.

If you auto-install the integration record, be aware that data specific to the first installation is not retained. The web services execution log from the previous installation is not retained. Additionally, the State field and Note field do not retain the values they had when the integration record was removed.

For details, see Auto-Installation of Integration Records.

Token-based Authentication Credentials and Accounts

Important: This content applies only to sandbox accounts for customers that have NA sandboxes.

If you are using Token-based Authentication (TBA) in your primary sandbox and production account, be aware of the following:

- The Token Key and Token Secret used for TBA are not part of information shared between your primary sandbox and production accounts.

- The Token Key and Token Secret are deleted in your primary sandbox account after each sandbox refresh, and you must regenerate them on your primary sandbox account. This has no impact on the Token Key and Token Secret used in your production account.
- The Consumer Key and Consumer Secret are shared between your primary sandbox and production accounts. If you change the Consumer Key or Consumer Secret in your primary sandbox account, auto install automatically changes them in your production account and can break your production account integrations.

Recommended practices:

- Create and test integration records in your primary sandbox and transfer them to your production account rather than creating separate integration records in your production account.
- Create a bundle that includes your integration records and TBA credentials on your primary sandbox and install the bundle to your production account. For details, see Bundle Installation of Integration Records.

Not recommended practices:

- Do not create new integration records to change the settings of an existing integration record. Instead, make changes on an existing integration record in your primary sandbox account and then install the record in your production account.
- If you decide after testing that you need to make changes to an integration record, do not create a new record in your production account with those changes. Instead, make the changes to the existing integration record in the sandbox account, test the changes, and then install the record on your production account.

Using Integration Records in Conjunction With SSO Calls

If you are using web services for single sign-on requests, be aware of the following:

- **Inbound** single sign-on requests are treated exactly as requests that authenticate through user credentials. For example:
  - If you are using the 2015.2 or a later WSDL, inbound single sign-on requests must include an application ID. To permit an application to authenticate by using inbound SSO, check the User Credentials box on the appropriate integration record. You can also use the integration record to block requests that include the corresponding application ID.
  - If your requests use the 2015.1 WSDL or earlier, you can optionally include an application ID. Requests that omit application IDs are tracked as part of the Default Web Services Integrations record. (You can use the Default integration record to view an execution log of these requests. Additionally, if you block the Default record, these requests are rejected.)
- With **outbound** single sign-on requests (SuiteSignOn requests), you do not have to send an application ID, regardless of which endpoint you are using.

Removing Integration Records

When an integration record is created in a NetSuite account by a user filling out the Integration form, two things happen: A record is added to, or installed, in that NetSuite account. Additionally, within the broader NetSuite database, a definition of that record is created. This definition can never be deleted. However, in some cases, the record can be uninstalled from the account in which it was created or the one to which it was distributed.

Further, you can always hide an integration record from the list view available at Setup > Integration > Manage Integrations.

For details, see the following topics:

- Uninstalling an Integration Record from a NetSuite Account
- Removing Installed Integration Records from View

Uninstalling an Integration Record from a NetSuite Account

An integration record can be uninstalled from a NetSuite account only in the following scenarios:

- If you install an integration record as part of a bundle, the record can be uninstalled from your account if you uninstall the bundle. Similarly, if the bundle owner removes the record from the bundle, you can remove the record when you update the bundle.
- If you create an integration record in a sandbox account and you do not propagate it to the production account, the record is removed when the sandbox is refreshed.

If you remove the integration record in either of the following scenarios, you can still auto-install the application later in certain circumstances. For details, see Auto-Installation of Integration Records.

Removing Installed Integration Records from View

Regardless of how an integration record was added to your account, you can hide it from view by blocking it.

To remove an integration record from view:

1. Navigate to Setup > Integration > Manage Integrations, and open the appropriate record for editing. Set the **State** field to **Blocked**. Note that making this change prevents the application from authenticating to NetSuite by using the application ID or consumer key associated with this record. For further details, see Blocking Web Services Requests. Click **Save**.
2. In the list view, make sure the **Show Inactives** box has been cleared. This configuration ensures that no blocked applications appear in the list view.

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 changed the value of the field, the system note also shows the field’s former value.

For more details, see the following topics:

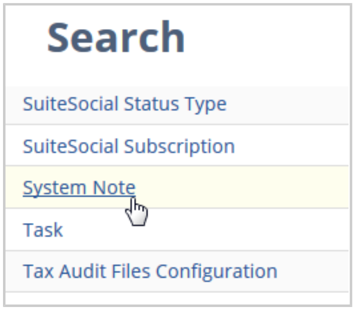
- Finding System Notes for an Integration Record
- Special Cases Related to System Notes Logging

For general details about working with system notes, see the help topic Searching System Notes.

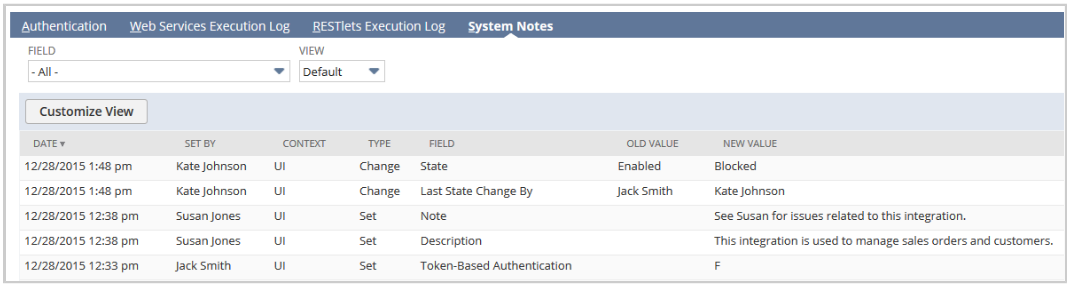
Finding System Notes for an Integration Record

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.



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

Special Cases Related to System Notes Logging

System notes for the integration record are similar to system notes for other types of records. However, be aware of special behaviors related to certain fields and situations, as follows:

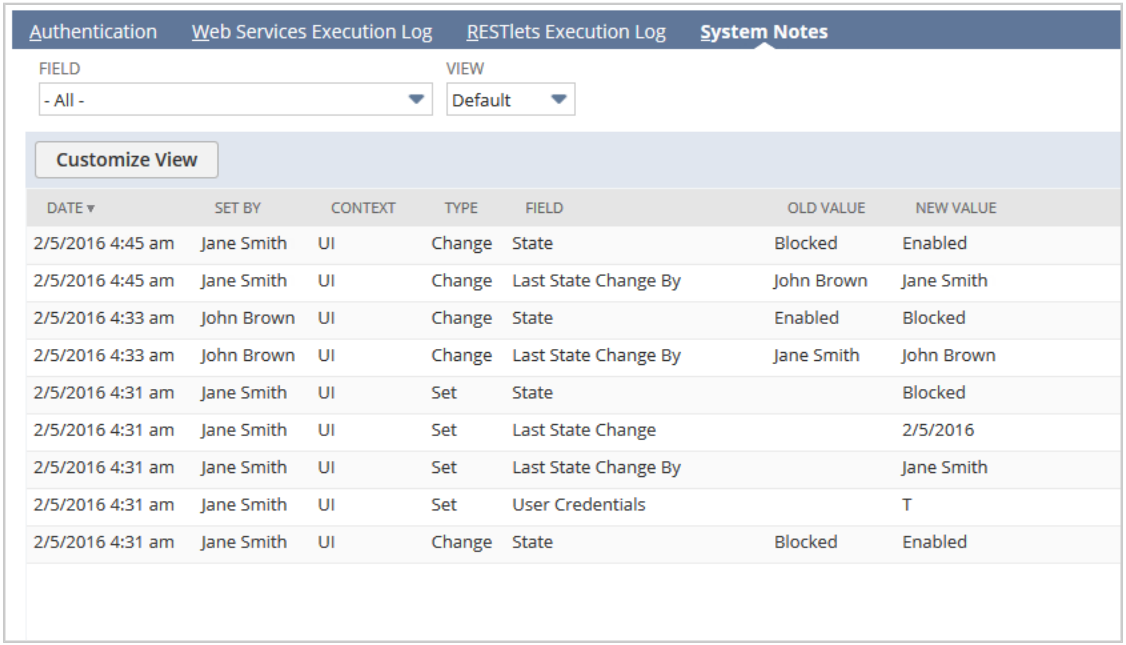
- Logging for the Last State Change Field
- Logging for the State Field
- Logging for the Reset Credentials Button
- Generation of an Application ID Is Not logged
- Logging for Records Shared by Sandbox and Production Accounts
- Notes Are Logged Only for Fields You Have Permission to Modify

Logging for the Last State Change Field

The Last State Change field shows the date of the last change to the **State** field. Changes to this field are reflected in the record's system notes. However, note the following: If the **State** field is changed multiple times in one day, only one system note is created to reflect modification to the **Last State Change** field. The reason is that this field shows only a date, not a time of day.

However, you can track the details — including the times — of all State changes, by the system notes logging for the **State** and **Last State Change By** fields: System notes are created to reflect all of each day's changes to these fields. Each of these system notes includes a date and time.

For example, the following screenshot shows system notes for a record that had its **State** changed several times on Feb. 5. The log includes only one change to the **Last State Change** field, but the details of each State change are logged through the other system notes.

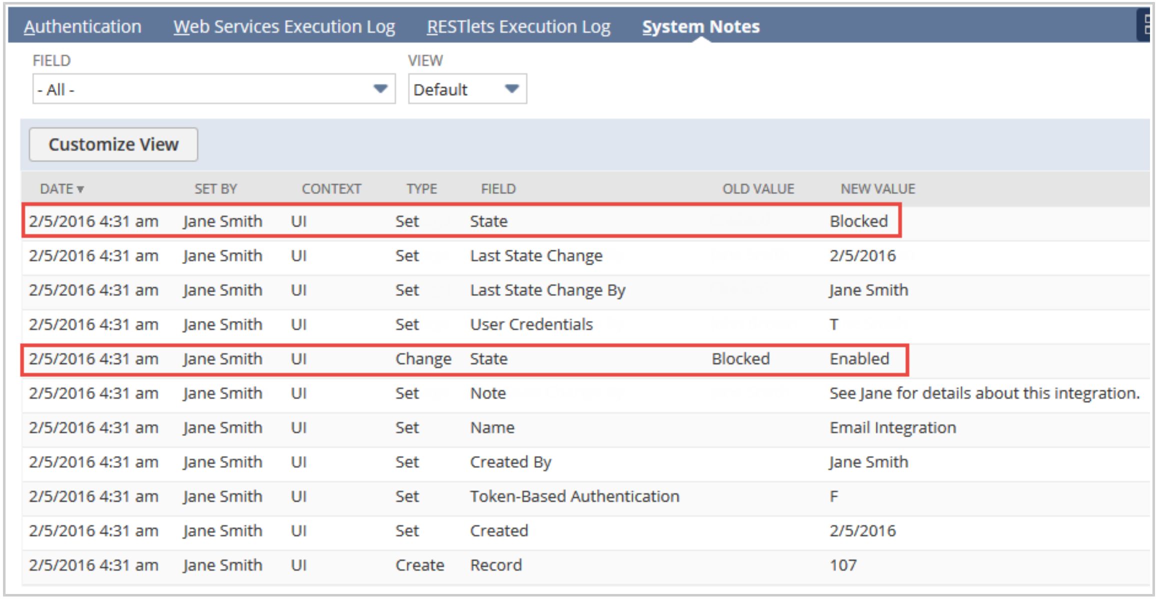


FIELD: - All -
VIEW: Default

DATE ▾	SET BY	CONTEXT	TYPE	FIELD	OLD VALUE	NEW VALUE
2/5/2016 4:45 am	Jane Smith	UI	Change	State	Blocked	Enabled
2/5/2016 4:45 am	Jane Smith	UI	Change	Last State Change By	John Brown	Jane Smith
2/5/2016 4:33 am	John Brown	UI	Change	State	Enabled	Blocked
2/5/2016 4:33 am	John Brown	UI	Change	Last State Change By	Jane Smith	John Brown
2/5/2016 4:31 am	Jane Smith	UI	Set	State		Blocked
2/5/2016 4:31 am	Jane Smith	UI	Set	Last State Change		2/5/2016
2/5/2016 4:31 am	Jane Smith	UI	Set	Last State Change By		Jane Smith
2/5/2016 4:31 am	Jane Smith	UI	Set	User Credentials		T
2/5/2016 4:31 am	Jane Smith	UI	Change	State	Blocked	Enabled

Logging for the State Field

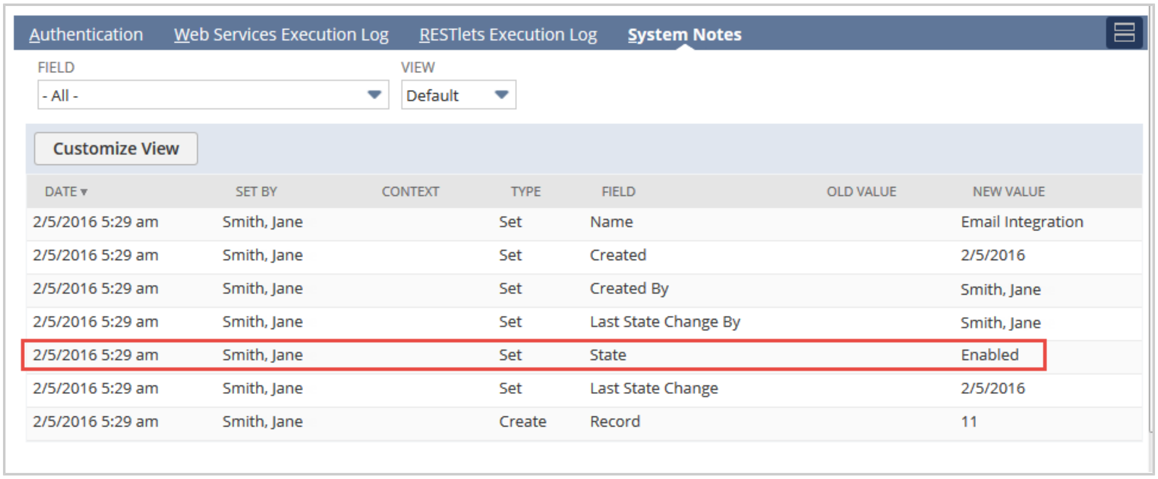
When you create an integration record and save it with its **State** set to the default of Enabled, the system logs two system notes in regard to this field: One note shows that the field initially had a value of Blocked. Another system note with the same timestamp shows that the field's value changed to Enabled.



FIELD: - All -
VIEW: Default

DATE ▾	SET BY	CONTEXT	TYPE	FIELD	OLD VALUE	NEW VALUE
2/5/2016 4:31 am	Jane Smith	UI	Set	State		Blocked
2/5/2016 4:31 am	Jane Smith	UI	Set	Last State Change		2/5/2016
2/5/2016 4:31 am	Jane Smith	UI	Set	Last State Change By		Jane Smith
2/5/2016 4:31 am	Jane Smith	UI	Set	User Credentials		T
2/5/2016 4:31 am	Jane Smith	UI	Change	State	Blocked	Enabled
2/5/2016 4:31 am	Jane Smith	UI	Set	Note		See Jane for details about this integration.
2/5/2016 4:31 am	Jane Smith	UI	Set	Name		Email Integration
2/5/2016 4:31 am	Jane Smith	UI	Set	Created By		Jane Smith
2/5/2016 4:31 am	Jane Smith	UI	Set	Token-Based Authentication		F
2/5/2016 4:31 am	Jane Smith	UI	Set	Created		2/5/2016
2/5/2016 4:31 am	Jane Smith	UI	Create	Record		107

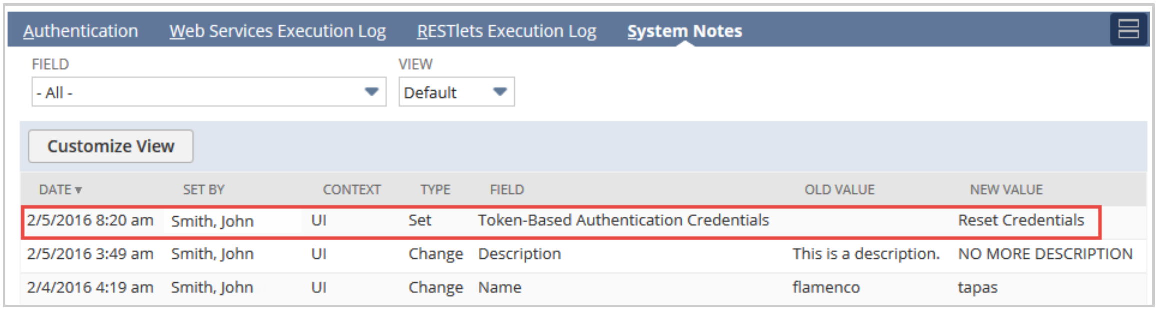
By contrast, if an integration record was installed through auto-installation or bundling, only one event is listed for the State field. This note shows that the field was saved with whatever value it had at the time of installation.



DATE ▾	SET BY	CONTEXT	TYPE	FIELD	OLD VALUE	NEW VALUE
2/5/2016 5:29 am	Smith, Jane		Set	Name		Email Integration
2/5/2016 5:29 am	Smith, Jane		Set	Created		2/5/2016
2/5/2016 5:29 am	Smith, Jane		Set	Created By		Smith, Jane
2/5/2016 5:29 am	Smith, Jane		Set	Last State Change By		Smith, Jane
2/5/2016 5:29 am	Smith, Jane		Set	State		Enabled
2/5/2016 5:29 am	Smith, Jane		Set	Last State Change		2/5/2016
2/5/2016 5:29 am	Smith, Jane		Create	Record		11

Logging for the Reset Credentials Button

If you own an integration record that has the Token-based Authentication option selected, you may at times need to reset the consumer key and secret. To regenerate these values, you open the record for editing and click the **Reset Credentials** button. This action is logged in the system notes as the setting of the Token-Based Authentication Credentials field. The system lists a new value of Reset Credentials.



DATE ▾	SET BY	CONTEXT	TYPE	FIELD	OLD VALUE	NEW VALUE
2/5/2016 8:20 am	Smith, John	UI	Set	Token-Based Authentication Credentials		Reset Credentials
2/5/2016 3:49 am	Smith, John	UI	Change	Description	This is a description.	NO MORE DESCRIPTION
2/4/2016 4:19 am	Smith, John	UI	Change	Name	flamenco	tapas

The actual credentials are not displayed in the system notes, just as they are not displayed anywhere on the record after being shown for the first time.

Generation of an Application ID Is Not logged

When you create a new integration record in your account, an application ID is automatically generated. However, this value is not one of those values listed in the system notes at the time you save the new record. The system notes contain no indication that a value for the field was set.

If you need to locate the application ID, it is visible on the integration record itself and in the integration record list view, but only by authorized users in the account that owns the record. Note that, although application IDs are not considered authentication credentials, NetSuite nevertheless recommends that application IDs be considered confidential.

Logging for Records Shared by Sandbox and Production Accounts

In some cases, an integration record might exist in both your production and sandbox accounts. For example, you could create a record in your sandbox and install it in your production account. In many such cases, the record is fully editable in both accounts. (For details on when this is possible, see Using Integration Records in Sandbox Accounts.) However, if you make a change in one account and the change is propagated to another account, be aware that a system note is created only in the account where the change was made.

Notes Are Logged Only for Fields You Have Permission to Modify

In general, if you install an integration record, system notes are created only for those fields that you are permitted to change. For that reason, the same record, when viewed in the account that owns it, contains system notes for a greater number of fields.

This distinction may be particularly noticeable if an account owner makes changes to fields that other accounts are not permitted to change. These updates are propagated to all accounts that have installed the record, but system notes are not created in the other accounts.

For details on how ownership of an integration record affects system notes for various fields, see the following table.

Field	Notes
Created By	When a record is created or installed, a system note is created to show the value of the **Created By** field. However, the way this field is populated varies, as follows: In the account where the record was created, the field's value identifies the person who created the record. If the record was installed, the field identifies the person who installed the record.
Description	System notes for the **Description** field exist only in the owner's account, even though changes are pushed to all accounts where the record has been installed.
Name	A system note is created for the **Name** field in an account that installs the record, even though that account does not have the ability to edit the field. In this sense, the Name field is unique among fields that are not modifiable by non-owning accounts. Note also that subsequent changes to the Name field by the account that owns the record are not propagated outside the account that owns the record, even though a bundle update may show a new name for the record.
Note	Data in the **Note** field is specific to the account where the record is installed. Therefore, a value for the Note field is never set during installation, and no system note is created for it at the time of installation. If you add a value for the field after installation, a system note is created.
Token-based Authentication	System notes for the **Token-based Authentication** field exist only in the owner's account, even though changes are pushed to all accounts where the record has been installed.
User Credentials	System notes for the **User Credentials** field exist only in the owner's account, even though changes are pushed to all accounts where the record has been installed.

Searching for Web Services Log Information

Beginning in 2016.2, you can access web services log information through searches in the NetSuite UI. You can set up results columns and filter data to return only the details you need. These searches support analysis of web services failures.

You can create web services log searches by going to Reports > New Search and selecting one of the following new search types:

- **Web Services Operations** — Use this type to search for and analyze information about failing web services requests and operations.
- **Web Services Record Processing** — Use this search type to search for and analyze information about failing web services records.

Each new search type includes join fields to the other search type, so the two search types can be joined for a more comprehensive web services log information search.

Note the following:

- Web services execution log items are retained for one month, so searches do not return data older than one month.
- Results for these searches are limited to log items created as of 2016.2; log items created prior to this release are not included.

Permissions for Accessing Web Services Searches

Three permissions have been added to NetSuite 2016.2 to for the web services search types in the UI.

- The Reports type Integration permission provides access to the **Integration** field in the Web Services Operations search.
- The Setup type Integration Application permission provides access to the **Request** and **Response** fields in the Web Services Operations search.
- The Setup type Set Up Web Services permission provides access to the Web Services Record Processing search.

Using the Web Services Operations Search Type

The following web services operation fields are available for search:

- **API Version** — The API version used to perform the operation.
- **Authentication Method** — The authentication method used by the operation.
- **Job ID** — The Job ID.
- **Time** — The date and time the operation was performed.
- **Duration** — The length of time (in seconds) it took to process the operation.
- **Operation** — The web services operation.
- **Operation Status** — The operation status; failed, finished, finished with errors, processing, or rejected.

- **User Email** — The email address associated with the account that performed the operation.
- **Records In Request** — The number of records in the request.
- **Records in Response** — The number of records in the response.
- **Failed Records** — The number of records that failed.
- **Service Type** — The type of service used: SYNC or ASYNC.
- **Integration** — The related integration record.
- **Finished Records** — The number of records completed for the selected operation. This column is available for advanced searches.
- **Request** — A link to the SOAP request. This column is available for advanced searches.
- **Response** — A link to the SOAP response. This column is available for advanced searches.

Using the Web Services Record Processing Search Type

The following web services record fields are available for search:

- **Duration** — The length of time (in seconds) it took to process the record.
- **Operation** — The operation performed on the record, either ADD, UPDATE, or DELETE.
- **Record Type** — The record type.
- **Sequence Number in Request** — When multiple records are processed during an operation, this field indicates the sequence number or processed record within the operation.
- **Status** — The record’s status, either Complete, Partially complete, or Failed.
- **Time** — The date and time when the record was processed.
- **Record** — The record that was processed. This column is available for advanced searches.

Roles and Permissions in Web Services

NetSuite provides many standard roles with predefined permissions. A role is a set of permissions that allows customers, vendors, partners and employees access to specific aspects of your data. Each role grants access at a certain level for each permission.

When logging in using web services you may provide a **role id** along with your **credentials**. The role id that you provide must have web services permissions, otherwise an **INSUFFICIENT_PERMISSION** error is returned. If no role id is provided, then the user's default role is used. If the default role does NOT have web services permissions, then a **ROLE_REQUIRED** fault is returned.

The following topics are provided in this section. They do not need to be read in order. However if you are new to NetSuite web services, it is recommended that you read each topic to understand how NetSuite roles and permissions apply in a web services context.

- Role and Permission Considerations When Developing in SuiteTalk
- Assigning the Web Services Permission to a Role
- Setting a Default Role for a Web Services User
- Setting a Web Services Only Role for a User
- Customer Center, Vendor Center, and Partner Center Roles
- Internal IDs Associated with Roles

Role and Permission Considerations When Developing in SuiteTalk

Due to SuiteTalk's reliance on the NetSuite role-based permissions, it is important for SuiteTalk developers to put that into considerations during the design phase to ensure smooth deployments.

It is common for developers to use the administrator role during development time because it gives them full permissions and access to all the records and operations. However, the target end users are likely to have less powerful roles, which may not have access to the data the SuiteTalk application requires.

Another role-related consideration is the preferred custom forms of some roles may not have access to certain fields or sublists that a SuiteTalk application requires. Hence the application's attempts to set those fields will result in permission errors.

The solution to these problems is to define a custom role and custom forms for the SuiteTalk application. The custom role should have the correct access permissions and operations permissions that the SuiteTalk application needs. The custom forms should give access to fields and sublists that are relevant to the SuiteTalk application. All SuiteTalk supported records have a `customForm` field for the application to reference specific custom forms.

Note: In 2016.2, a permission has been added to the Role page on the Permissions > Setup subtab. If the Control SuiteScript and Workflow Triggers in Web Services Request permission is selected, users cannot change the setting for scripts and workflow triggers on individual web services requests. For users who do not have permission to disable scripts, the global setting for the account applies for all of their requests.

Important: When testing SuiteTalk applications, you should do so using the role(s) of your intended users(s), in addition to the administrator role, to catch permission-related defects.

Assigning the Web Services Permission to a Role

All standard NetSuite roles have web services permissions by default. For security reasons, it is recommended that you restrict permissions levels and access allowing only the most restricted permissions necessary to perform a particular set of operations.

For non-standard or custom roles, use these steps to assign the web services permission to the role.

To assign the Web Services permission to a role:

1. Go to Setup > Users/Roles > Manage Roles.
2. Locate the role you want to modify. Click the corresponding **Edit** or **Customize** link.
3. Click the **Setup** subtab.
4. In the **Permissions** dropdown list, select **Web Services**.
5. In the Level dropdown list, select **Full**.

   Note: Users with a Web Services permission level other than Full (View, Create, Edit) cannot log in to web services. The Full level is required. Also note that the Web Services permission does not provide access to the Web Services Usage Log or to integration records; only administrators can access these pages. For details on the Web Services Usage Log, see Using the Web Services Usage Log. For details on working with integration records, see Web Services Security.

6. Click **Add**.
7. Click **Save**.

Note also:

- If you are building an integrated application, it is best to create a new role or customize an existing role and grant the minimum set of permissions that are necessary for the client to carry out its functions. It is not recommended that users are granted the Full Access role or that a user should be assigned administrator privileges in your web services.
- If your role has permission to view credit card data on the user interface, you can also retrieve this information through web services calls. This is beneficial to integrated applications that use an external credit card processing profile. Based on your role, you may be able to retrieve the credit card on file for your customers.
- In order for a role to use the getDeleted operation, the role has to have both the Deleted Records and the Web Services permissions. Note that users who have the Deleted Records permission can access results about any type of record that was deleted, even if they do not have permission to create or modify that record type. They also have permission to use the Deleted Record search type, unless the Web Services Only option has been selected for the role.

Setting a Default Role for a Web Services User

You can specify a default role for any user making web services requests. The permissions for the default role are determined as follows:

- First, any role specified in the Passport object of the request is used. The role defined here must be a valid role contained in the employee record of the specific user. (For information on the Passport object, see login. The Passport object is defined in the platformCore XSD.)
- If a role preference is not set at the request level, then any default web services role as defined in the Web Services Preference page for the specific user is used.

  Only one default web services role can be assigned per user and only roles that contain the web services permission can be specified as a default web services role. Note that the user may be assigned a different role than those specified in their employee record. In other words, a user may have greater or lesser permissions using web services as compared to the UI.
- If neither the request nor the web services default role is set, then the user's default UI role is used, provided it has the web services permission.

Note: All standard roles have the web services permission by default when the web services feature is enabled. Custom roles, however, must be explicitly set to have web services permissions.

To set a specific default role for a web services user:

1. Click Setup > Integration > Web Services Preferences.
2. Select the desired user from the **Name** dropdown list.
3. Select the default role to use for web services requests for this user.

   The internal ID for the selected role automatically populates the **ID** field.
4. Click **Done**.
5. Click **Save**.

Setting a Web Services Only Role for a User

In NetSuite you can designate a user's role as **Web Services Only**. When a user logs in with a role that has been designated as Web Services Only, validation is performed to ensure that the user is logging in through web services and not through the UI.

Note: Your account must have the web services feature enabled for the Web Services Only check box to appear. See Enabling the Web Services Feature for steps on enabling this feature.

The Web Services Only role increases the security of an integrated application by prohibiting a UI user from accessing the system with permissions and privileges that are specifically created for a web services applications. For example, you may have a web services application that requires certain employees to have write access to several records. However, you want to prohibit the employees from being able to edit these records directly from within the NetSuite UI. If you assign the Web Services Only role to specified employees, the employees can log in to NetSuite and access the application through web services, however, the employees cannot switch to their other roles within the system and write/edit/delete these any data-sensitive records.

Important: The Web Services Only role does not appear in the Change Role dropdown list. Therefore, users cannot change their roles from their original UI login role (A/P clerk, for example) to their Web Services Only role from within the UI.

To designate a role as Web Services Only:

1. Click Setup > Users/Roles > Manage Roles.
2. On the Manage Roles list page, locate the role you want to set as Web Services Only.
3. Click the corresponding **Edit** or **Customize** link.
4. Check the **Web Services Only Role** box.
5. Click **Save**.

When to Set the Web Services Only Role

A role should not be designated as **Web Services Only** until the developers building and testing the integrated application have completed the application. Waiting to designate a role as **Web Services Only** allows developers to go back and forth during design and development time to test the permissions for the role that is designed specifically for an integrated application. After the development and testing is complete, the developer can set the **Web Services Only** role to TRUE for a specified role to prevent users with this role access to the UI with this set of permissions and privileges.

Note: External roles such as Customer Center, Partner Center, Advanced Partner Center, Vendor Center, and Employee Center should not be customized to have Web Services Only permissions.

Customer Center, Vendor Center, and Partner Center Roles

The Customer Center, Vendor Center, and Partner Center roles have implicit web services permissions. This allows integration with an externally hosted website where a client can execute any task available under the center-specific role through web services. For example, the client could login and submit an order on behalf of the customer.

Note: It is not recommended that you customize Customer Center, Partner Center, or Partner Center Roles to have only Web Services Only permissions. For information on the Web Services Only role, see Setting a Web Services Only Role for a User.

If you choose you can remove web services permissions from the Customer Center, Vendor Center, and Partner Center roles.

To remove web services permissions:

1. Go to Setup > Users/Roles > Manage Roles.
2. Locate the role you want to modify.
3. Click the corresponding **Edit** or **Customize** link.
4. Click the **Setup** subtab.

5. In the **Permissions** dropdown list, click the **Web Services** line to make it editable. The page updates to include several buttons below the line.
6. Click the **Remove** button.
7. Click **Save**.

Internal IDs Associated with Roles

The table in this section lists the standard NetSuite roles and the associated internal ID values. You can use these internal ID values in the Passport object, which is used for the login operation. For information on the Passport object and the login operation, see login.

If you have the Show Internal IDs preference on, you can look up the internal ID of a role by going to Setup > Users/Roles > Manage Roles. The role ID appears in the Internal ID column. For instructions on setting the Show Internal IDs preference, see Setting the Internal ID Preference.

Note: In addition to the NetSuite standard roles, there may exist custom roles that have been created by your organization. Custom roles are assigned internal IDs sequentially (starting with 1001).

ID	Is Settable	Role
1	Y	Accountant
2	Y	Accountant (Reviewer)
3	Y	Administrator
4	Y	A/P Clerk
5	Y	A/R Clerk
6	Y	Bookkeeper
7	Y	CEO (Hands Off)
8	Y	CEO
9	Y	Sales Manager
10	Y	Sales Person
11	Y	Store Manager
12	Y	Support Manager
13	Y	Support Person
14	Y	Customer Center
15	Y	Employee Center
16	Y	Vendor Center
17	**N**	Shopper
18	Y	Full Access
19	Y	Warehouse Manager
20	Y	Payroll Manager
21	**N**	Partner Center
22	Y	Intranet Manager

ID	Is Settable	Role
23	Y	Marketing Manager
24	Y	Marketing Assistant
25	Y	System Administrator
26	Y	Sales Administrator
27	Y	Support Administrator
28	Y	Marketing Administrator
29	**N**	Advanced Partner Center
30	Y	NetSuite Support Center
31	**N**	Online Form User

Records, Fields, Forms, and Sublists in Web Services

The following topics are covered in this section. These topics do not need to be read in order, however, if you are new to SuiteTalk development, you will need to review each section to fully understand the SuiteTalk API.

- Records in Web Services
- Fields in Web Services
- Forms in Web Services
- Sublists in Web Services

Records in Web Services

Most standard NetSuite records are supported by SuiteTalk. The list of supported records spans all areas of the NetSuite application from ERP to CRM to customization. For a list of records that are supported in SuiteTalk, see the help topic Web Services Supported Records.

In the SuiteTalk API, the Record class is the abstract super-class of all supported records. A supported record is always a concrete sub-class of Record. Due to the neutrality required to be language agnostic, the SuiteTalk classes inheritance chain remains simplistic and does not implement language-specific object-oriented concepts such as multiple inheritance and interfaces.

NetSuite records are divided into the following broad categories:

- Record Types
- Search Records
- Subrecords

A record's standard body fields are its attributes, for example: `Customer.email`, `SalesOrder.salesRep`. Composite attributes such as line items or sublists are structured as complex objects that contain arrays, for example: `SalesOrder.itemList`, `CalendarEvent.attendeeList`. Custom fields within a record (if available) are also structured as composite attributes, for example: `Contact.customFieldList`.

Note: A record element that ends with **List** is generally a sublist. (The exception is the customFieldList element, which represents custom fields on the record.) For information on working with sublists in web services, see Sublists in Web Services. For more general information on sublists in NetSuite, see the help topic What is a Sublist?

In addition to standard records, SuiteTalk also supports custom objects and their metadata (see the help topic Customization for more details). Using the getCustomizationId SuiteTalk API, an application external to NetSuite can query a NetSuite account to obtain metadata about the custom objects that have been implemented in the account. This lets you build and ship generic applications that will work with any account. For example, a SuiteTalk point-of-sale application can be designed to determine (during runtime) all the custom fields applied to a NetSuite CashSale record, so that it can then import CashSale records with the necessary custom fields set.

Record Types

A NetSuite record type typically represents specific business functions such as creating a customer or updating an opportunity. For a list of all supported record types, see the help topic Web Services Supported Records.

Search Records

Search record types encapsulate the available search criteria for a specific NetSuite record type. A NetSuite search record is defined as a top-level record that is used in the request portion of a search operation. Any field defined within a search record must be of one of the following logical types.

Type	Description
String	Corresponds to the SearchStringField type
Int	Corresponds to the SearchTextNumberField type
Double	Corresponds to the SearchDoubleField type
Boolean	Corresponds to the SearchBooleanField type
Datetime	Corresponds to the SearchDateField type
MultiSelectRef	Corresponds to the SearchMultiSelectField type
MultiSelectEnum	Corresponds to the SearchEnumMultiSelectField type

For details on using the search operation, see search. Also see the following topics, which describe all search types that can be used when constructing web services searches.

- Which SuiteTalk objects are used in a basic search?
- Which SuiteTalk objects are used in a joined search?
- Which SuiteTalk objects are used in advanced search?

Subrecords

A subrecord is similar to a record. You can take many of the same actions with a subrecord as you can with a record. However, like a sublist, an instance of a subrecord is always contained within a parent record. The first subrecord, inventory detail, was exposed to web services in the 2011.2 endpoint.

Any add, get, update, delete, or search operation on a subrecord must be performed within the context of an operation on its parent record. For example, if you want to update inventory detail data for a purchase order transaction, you must update the purchase order record itself. You cannot do an independent update of the inventory detail object. This limitation applies to all operations on a subrecord.

The following preferences apply to subrecords:

- For an add operation, you can put the subrecord in the nullFieldList to null out the subrecord.
- If a subrecord is not in the nullFieldList, its replaceAll attribute determines whether the subrecord is added or updated during an add operation on its parent record. If this attribute is set to true, the subrecord is deleted and a new one is added. If this preference is set to false, an update of the subrecord is attempted.

Using Internal IDs, External IDs, and References

Each record in NetSuite can be uniquely identified by its record type in combination with either an external ID or a system-generated NetSuite internal ID.

Internal and external IDs are NOT unique across all record types. Therefore, to perform an operation on an existing record, you need two pieces of data: the record type and either the internal ID or the external ID.

References are implemented through the RecordRef type, which is defined in the core XSD.

The RecordRef type is described by three attributes — the internal ID, external ID, and the type:

```
<complexType name="RecordRef">
<complexContent>
   <extension base="platformCore:BaseRef">
      <attribute name="internalId" type="xsd:string"/>
      <attribute name="externalId" type="xsd:string"/>
      <attribute name="type" type="platformCoreTyp:RecordType"/>
   </extension>
</complexType>
```

Important: When referencing records in an update operation, you must provide either the internal ID or the external ID, but not both. If both are provided, the internal ID is used to locate the record and the external ID is ignored. Additionally, the external ID attribute is case insensitive.

Uniqueness of Internal and External IDs

Internal and external IDs must be unique not just within a specific record type, but also within certain record groups. These groups consist of multiple record types. For details, see Shared Internal and External IDs.

Internal IDs are not reused in the system. After an internal ID has been assigned, if the associated record is subsequently deleted, the ID is not reused.

Note also that an external ID exists only if you provided it, either at the time of record creation or during an update. For more on external IDs, see External IDs Overview.

Internal IDs Do Not Indicate When Record Was Created

The assignment of internal IDs to new records is sequential. However, the sequence of internal IDs does not indicate when or in which order records were added. This is true even within specific record groups where internal IDs are unique. Therefore, you should not conclude that records with lower internal IDs were added first, whereas those with higher internal IDs were added last.

Consider the following example. All records belonging to the same high-level group share the same ID pool. For example, sales orders and invoices both belong to the group of transaction records. When you create a sales order and it is assigned 1234 as the internal ID, you should not assume that the internal ID of the next sales order will be 1235. If an invoice or another transaction record is created right after sales order 1234, that record might have 1235 as internal ID. Additionally, there can be gaps between internal IDs. Two sales orders created a few minutes' apart might have a significant gap between their internal IDs.

For custom applications that rely on the chronological aspect of when data was entered, you should use the Date Modified or Date Created fields, as they are guaranteed to be chronological. Additionally, for transactions with auto-generated numbers, the sequence number (for example, “tranid”) is guaranteed to be sequential based on the date transactions were entered. The internal ID of a record should never be used to determine when data was entered, and you should not build any functionality based on the sequence of internal IDs.

Internal IDs for NetSuite Record Types

Like individual records, each record *type* also has an internal ID. In certain situations, you must use this ID to specify which type of record you are referencing. By contrast, in some cases, you can use a string to reference the record type (for example, when using RecordRef). But in other cases you must use the internal ID, or typeId (for example, when using ListOrRecordRef or CustomRecordRef). The table below lists the NetSuite record types and their associated typeIds.

Custom fields of the type MultiSelectCustomFieldRef on custom records also reference these values, because they contain an array of ListOrRecordRef types. For details on working with custom records, see the help topic Custom Record.

Record Type	typeId
Account	-112
Accounting Period	-105
Bin	-242
Call	-22
Campaign	-24
Case	-23
Class	-101
Competitor	-108
Contact	-6
Customer	-2
Customer Category	-109
Department	-102
Email Template	-120
Employee	-4
Employee Type	-111
Entity Status	-104
Event	-20
Issue	-26
Item	-10
Item Type	-106
Job (Project)	-7
Location	-103
Module	-116
Opportunity	-31
Partner	-5
Product	-115
Product Build	-114
Product Version	-113
Project (Job)	-7
Role	-118
Saved Search	-119
Subsidiary	-117

Record Type	typeId
Task	-21
Transaction	-30
Transaction Type	-100
Vendor	-3
Vendor Category	-110

External IDs Overview

The externalId attribute of a RecordRef provides a means to reference an object by its foreign key in an external database.

You can set the externalID attribute during an add or update operation. External IDs are useful in the following situations:

- **Maintaining client ID relationships**

  In cases where a client application already maintains references between records, set the externalId attribute for each record during imports. In subsequent API calls, you can then reference associated records by the known external ID.
- **Establishing relationships during a single import operation**

  For example, suppose you want to import customer records with references to sales reps into NetSuite. If no external ID is used, you would need to import the customer records, determine the IDs of the related sales reps' employee records, and then re-import the customer records with the sales reps ID references. By providing an external ID, you can import the customer records in a single API call using the external ID references to the sales reps.

To prevent duplicate records, you should use external IDs and the upsert and upsertList operations to add records to NetSuite.

Most NetSuite record types support the use of external ID, but not all do. The following list identifies the record types that do not support external ID. For all other records, you should assume that external ID is supported. When in doubt, refer to the appropriate page in the SuiteTalk Schema Browser. On each page describing a record type, view the Attributes table to see whether external ID is supported.

The record types that do not support external ID are:

- Accounting Period
- Budget Category
- CRM Custom Field
- Currency Rate
- Custom List
- Custom Record Custom Field
- Custom Record Type
- Entity Custom Field
- Gift Certificate
- Item Custom Field
- Item Number Custom Field
- Item Option Custom Field
- Landed Cost

- Other Custom Field
- State
- Transaction Body Custom Field
- Transaction Column Custom Field

Guidelines for External IDs

Consider the following guidelines for external IDs:

- External IDs must be unique not just within a single record type, but also within certain record groups. For details about record types and record groups, see Shared Internal and External IDs.
- External IDs can be updated through CSV import or web services. Therefore, it is recommended that your organization use a single approach for maintaining external IDs, so that external IDs are not unknowingly updated using two separate methods.

  Although records of a particular type may be used in multiple integration scenarios, each record instance can only have a single external ID value. To maintain data integrity, only a single integrated application can set and update external ID values for each record type. External ID values for all records of a particular type must all come from the same external application.
- External IDs are case insensitive in web services references. For details, see Using Internal IDs, External IDs, and References.

Shared Internal and External IDs

System-generated internal IDs and custom external IDs are shared among records belonging to the same high-level group. Therefore, when referencing a record using RecordRef, providing the system internal ID or custom external ID without specifying the record type is sufficient to uniquely identify the record within a specific group.

External IDs must be unique across the whole record group, not just per record type. For example, an external ID for a customer must not be reused for any other customer or for any other entity record, including contacts, employees, groups, partners, or vendors. To avoid clashes within record groups, use a consistent format for constructing your external IDs.

The following table provides the list of these high-level groups and the records belonging to each group.

Record Group	Record Types
Account	Account; Tax Account
Allocation Schedule	Allocation Schedule; Intercompany Allocation Schedule
Entity	Contact; Customer; Employee; Entity; Entity Group; Job; Lead; Other Name; Partner; Project Template; Prospect; Vendor
Entity Status	Customer Status; Job Status
Event	Activity; Campaign; Case; Event; Issue; Manufacturing Operation Task; Phone Call; Project Task; Resource Allocation; Sales Campaign; Solution; Task
Inbound Shipment	Bulk Ownership Transfer; Inbound Shipment; Receive Inbound Shipment
Inventory Item	Assembly/Bill of Materials; Description Item; Discount Item; Download Item; Expense Item; Gift Certificate Item; Inventory Part; Item; Item Group; Kit Item;

Record Group	Record Types
	Lot Numbered Assembly/Bill of Materials; Lot Numbered Inventory Item; Markup Item; Non-Inventory Part; Non-inventory Item for Purchase; Non-inventory Item for Resale; Non-inventory Item for Sale; Other Charge Item; Other Charge Item for Purchase; Other Charge Item for Resale; Other Charge Item for Sale; Payment Item; Serialized Assembly/Bill of Materials; Serialized Inventory Item; Service; Service Item for Purchase; Service Item for Resale; Service Item for Sale; Subscription Plan; Subtotal Item; Tax Group
Payroll Batch	Payroll Batch; Payroll Batch - Add Employees
Project Revenue Rule	Fixed Amount Project Revenue Rule; Labor Based Project Revenue Rule; Percent Complete Project Revenue Rule
Revenue Recognition Schedule	Amortization Schedule; Amortization Template; Revenue Recognition Schedule; Revenue Recognition Template
Timeline	Manufacturing Planned Time; Time; Time Entry
Transactions	Advanced Intercompany Journal Entry; Assembly Build; Assembly UnBuild; Bin Transfer; Bin Worksheet; Blanket Purchase Order; Cash Refund; Cash Sale; Check; Commission; Credit Card Charge; Credit Card Refund; Credit Memo; Custom Transaction; Customer Deposit; Customer Payment; Customer Payment Authorization; Customer Refund; Deposit; DepositApplication; Estimate; Expense Report; Fulfillment Request; FxReval; GL Impact Adjustment; Intercompany Journal Entry; Intercompany Transfer Order; Inventory Adjustment; Inventory Cost Revaluation; Inventory Count; Inventory Status Change; Inventory Transfer; Inventory Worksheet; Invoice; Item Fulfillment; Item Receipt; Journal Entry; Opportunity; Ownership Transfer; Partner Commission; Paycheck; Paycheck Journal; Purchase Contract; Purchase Order; Request For Quote; Requisition; Return Authorization; Revenue Arrangement; Revenue Commitment; Revenue Commitment Reversal; Revenue Contract; Sales Order; Statement Charge; Statistical Journal Entry; Store Pickup Fulfillment; Transaction; Transfer; Transfer Order; Vendor Bill; Vendor Credit; Vendor Payment; Vendor Request For Quote; Vendor Return Authorization; Work Order; Work Order Close; Work Order Completion; Work Order Issue

Fields in Web Services

NetSuite records contain standard body fields and custom fields. Standard fields are those that come with all NetSuite accounts. In web services, a record's standard body fields are its attributes, for example: `Customer.email, SalesOrder.salesRep`. Standard fields must be of one of the following logical types.

Type	Description
String	Corresponds to the xsd:string type in the XML Schema
Int	Corresponds to the xsd:int type in the XML Schema
Double	Corresponds to the xsd:double in the XML Schema
Boolean	Corresponds to the xsd:boolean type in the XML Schema and has valid values of true or false. If not explicitly set to either true or false, then set as false.
Datetime	Corresponds to the xsd:dateTime type in the XML Schema which conforms to the ISO 8601 standard.

Type	Description
RecordRef	Corresponds to the RecordRef type in the XML Schema. References an nsKey value for any other record in the system including system defined constants that are controlled by the system.
Enum	Corresponds to a specific type defined as an enum in the XSD that represents system constants that are also available in the UI.
WsEnum	Corresponds to a specific type defined as an enum in the XSD that represents system constants that are NOT available in the UI.
List	A List references a type that is a list and should be explicitly defined in the in the XML Schema as a type. A list can either be null (if it is an optional field), or it must contain at least one entry unless otherwise noted.

Custom fields are those that have been created by NetSuite users to customize their accounts. Custom fields can be added to an account using SuiteBuilder point-and-click customization tools. They can also be added through the SuiteTalk add operation. Custom fields must be one of the types defined in Custom Field Types.

On records, custom fields are contained in the customFieldList property. For more details, see CustomFieldList.

Note: If you are getting unusual results in your web services queries, it may be that a standard field has been customized.

Consider the following general guidelines when working with both standard and custom fields.

- If a **standard** field is set as mandatory in a custom form, the "requiredness" of the field is honored in web services. If you do not provide value in your web services request for a field that is set as mandatory in a custom form, an error message is returned. For more information on NetSuite form types, see the help topic Custom Forms.
- Customizations made to **standard** fields are honored in web services. For example, if a standard field is set as disabled, it is not settable through web services, either. If you try to set such a field through web services, an error message is returned.
- If a **standard** field is set to NOT show in the UI, it is not settable through web services, either. If you try to set such a field through web services, an error message is returned.
- **Custom** display only (inline) and disabled fields are not settable through web services because these are NOT settable through the UI. If you provide a value in your web services request for such a field, an error message is returned. For information on field display types, see the help topic Setting Display Options for Custom Fields.
- The behavior of **custom** hidden fields varies depending on the endpoint being used. If you are using any endpoint since the 2011.2 endpoint (including 2011.2), the get operation does not return the value of custom hidden fields. If you are using an older endpoint, these values are gettable. By comparison, in the UI, these values cannot be seen.

  If you are using the 2011.1 endpoint or earlier, the value of custom hidden fields is returned regardless of the setting of the ignoreReadOnlyFields preference.

  Custom hidden fields cannot be set, either in the user interface or using web services, regardless of the endpoint.
- Defaulted fields that are set to blank on a web services update will stay blank on update.
- The bodyFieldsOnly request-level preference affects the way formulas on forms are processed. As a result, if the values of custom fields are calculated using formulas, the search might return incorrect results. If the calculated values of formulas are incorrect, set the bodyFieldsOnly preference to false to make sure the correct values are returned. For more information, see Request-Level Preferences.

- If custom fields are not returned in search results, use the login and logout operations to create a new session after the searched record type is changed or updated, for example, if a field is added to the record type.

> **Note:** If you are unfamiliar with NetSuite custom fields, it is recommended that you see the help topic Custom Fields. This section describes the purpose and general characteristics of each custom field type, which will help you when working with the SuiteTalk customization API (described in Customization).

CustomFieldList

Most record types that are exposed through web services can have custom fields. Custom fields are contained in the customFieldList property on each record. The customFieldList property is an array of CustomFieldRef.

> **Note:** The Disable Mandatory Custom Field Validation preference determines whether a required custom field with no data provided throws an error, or is accepted as a null value. For more information on this preferences, see Company-Wide Preferences.

Field Name	Type	Req.	Default	Notes
customField	varies	Yes		Value of the custom field. Points to a type of CustomFieldRef in the XML Schema which is an abstract type.
internalID	string	Yes		The field instance internal ID
recType	string	No		The record type id
xsi:type	xsi:type	Yes		This is a field that is automatically implemented by the XML Schema. The value should represent the concrete custom field type.

The following is an example that contains an excerpt of the SOAP body for a list of custom fields. It shows that the record associated with the SOAP contains four custom fields, all of which are referenced in the customFieldList property on the record.

```
<listRel:customFieldList xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"
>
   <platformCore:customField internalId="72" scriptId="custentity_franchiseeabn_on_customerrec"
 xsi:type="platformCore:StringCustomFieldRef">
      <platformCore:value>{partner.vatregnumber}</platformCore:value>
   </platformCore:customField>
   <platformCore:customField internalId="424" scriptId="custentity_score" xsi:type="platformCor
e:LongCustomFieldRef">
      <platformCore:value>25</platformCore:value>
   </platformCore:customField>
   <platformCore:customField internalId="324" scriptId="custentity_expertise_level" xsi:type="p
latformCore:SelectCustomFieldRef">
     <platformCore:value internalId="2">
     <platformCore:name>Medium</platformCore:name>
     </platformCore:value>
   </platformCore:customField>
   <platformCore:customField internalId="57" scriptId="custentity_633637_bsubmit" xsi:type="pla
```

```
tformCore:BooleanCustomFieldRef">
     <platformCore:value>false</platformCore:value>
   </platformCore:customField>
</listRel:customFieldList>
```

Setting Custom Fields to NULL

Custom fields can only be set to NULL by submitting the field in nullFieldList. For example, to set a custom field on a customer record to null, submit the following SOAP request, where custEntity9 is the custom field ID and 373 is the specific instance of the customer record:

```
<soap:Body>
   <platformMsgs:update>
      <platformMsgs:record internalId="373" xsi:type="listRel:Customer">
         <platformCore:nullFieldList xsi:type="platformCore:NullField">
            <platformCore:name>custEntity9</platformCore:name>
         </platformCore:nullFieldList>
      </platformMsgs:record>
   </platformMsgs:update>
</soap:Body>
```

Note: You cannot set the Custom Form field to NULL. (This field is available on transaction and entry forms to indicate the form that should be used.) Any request to set it this field to NULL with nullFieldList will be ignored.

Sample .NET Code

```
Customer cust = new Customer();
cust.internalId = “373”;
cust.nullFieldList = new String[]{"custentity9"};
_service.update(cust);
```

Sample Java Code

```
Customer cust = new Customer();
cust.setInternalId(“373”);
NullField nfl = new NullField(new String[]{"custentity9"});
cust.setNullFieldList(nfl);
port.update(cust);
```

Custom Fields and Joined Searches

When a custom field is used to create a parent/child relationship between two existing records, the resultant joined searches available through the NetSuite UI are not supported in web services.

CustomFieldLists for Setting Custom Segment Values

Custom segments are custom classification fields similar to class, department, and location. When the Custom Segments feature is enabled, an administrator working in the UI can create custom segments, define possible values for each segment, and apply the segments to specific record types. Each custom segment appears as a field on instances of record types where the segment was applied. Users can then classify records by selecting the appropriate value for each segment.

When working with records in web services, you can in many cases set values for segments that may appear as fields on those records. Specifically, if the record type's exposure includes a CustomFieldList, you can set segment values on instances of that record type. To set values, you define a CustomFieldList for the record instance. You can then use the list to set values for one or more segments.

For more details on setting values for custom segments on records, see the following:

- Custom Segment Types
- Identifying the Correct Script ID for a Custom Segment
- Identifying a Custom Segment Value
- Example: Setting a List/Record Segment Value
- Example: Setting Values for Multiple Segments

Custom Segment Types

Every custom segment must have a type. The way you set a value for a segment in web services varies depending on the segment's type.

A segment can be either of two types, as described in the following table.

Type	Description	To represent in web services, use:
List/Record	Lets the user save only one selection for the segment	SelectCustomFieldRef
Multiple Select	Lets the user save multiple selections for the segment	MultiSelectCustomFieldRef

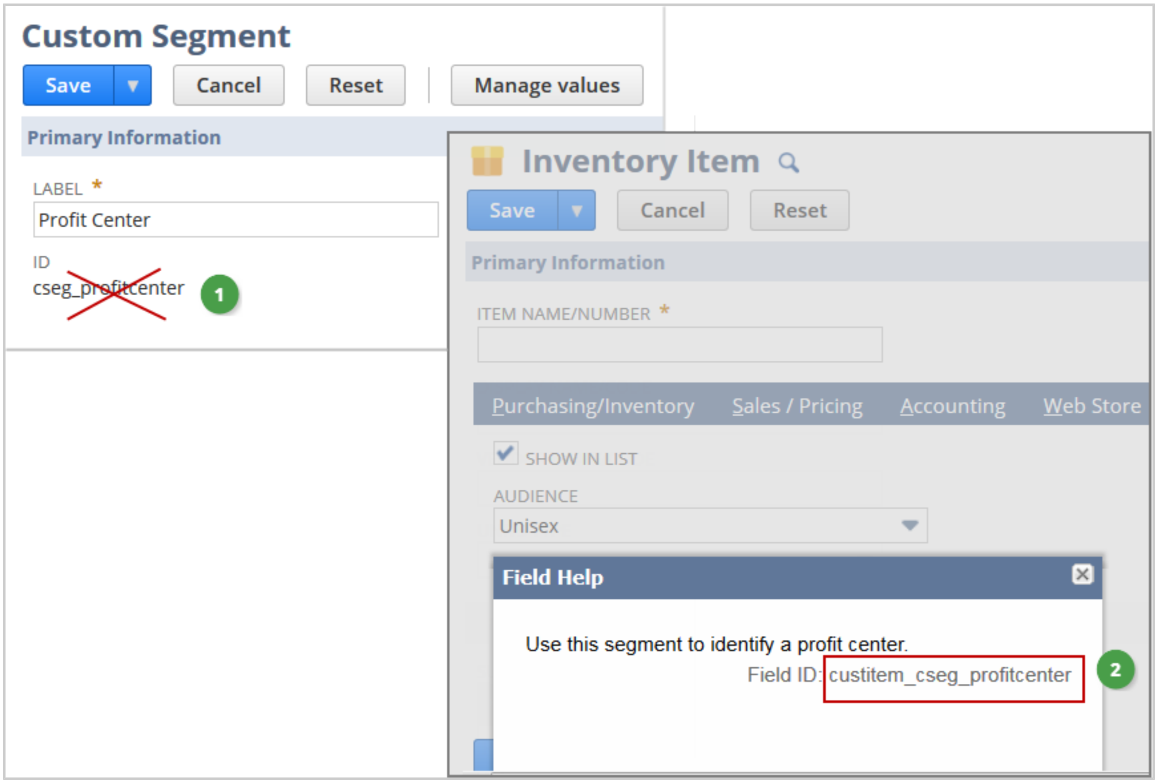
Both SelectCustomFieldRef and MultiSelectCustomFieldRef are represented by the CustomFieldRef abstract class. You can include both of these types of objects in a single CustomFieldList. For examples, see Example: Setting a List/Record Segment Value and Example: Setting Values for Multiple Segments.

Identifying the Correct Script ID for a Custom Segment

When using web services to set a value for a custom segment, you use the CustomFieldRef abstract class. As part of this approach, you use the CustomFieldRef's scriptId field to identify the exact segment for which you want to set a value.

Be aware that when you set a value for the scriptId field, you must use the ID as it appears on an instance of the record type that you are working with. Do not use the ID that appears on the custom segment record. On the custom segment record, the ID has a value that begins with the prefix cseg. On the record instance where the segment appears as a field, the ID has an additional prefix that corresponds with the category of the record type. You must use this longer ID that is specific to the record type category.

For example, on the custom segment record, the ID might have a value like cseg_profitcenter. If the segment has been applied to the iventory item record type, the segment ID on the inventory item record is custitem_cseg_profitcenter. If the same segment has been applied to a customer record, the ID on the customer record is custentity_cseg_profitcenter. Use the ID as it appears on the specific record type that you are working with.



Use this segment to identify a profit center.
Field ID: custitem_cseg_profitcenter

Callout	Description
1	Do not use this ID.
2	Use this type of ID.

To identify the ID for a segment:

1. If you do not have the Show Internal Ids preference set to true, enable it by completing the following steps:
    1. In NetSuite, point to the Home icon, and click **Set Preferences**.
    2. On the **Set Preferences** page, check the **Show Internal IDs** box.
    3. Click **Save**.
2. Navigate to an instance of the appropriate record type.
3. Locate the segment field. On standard forms, custom segments are listed on the **Custom** subtab.
4. Locate the appropriate segment field. Click on its label to display a popup box that includes the field’s ID.

Identifying a Custom Segment Value

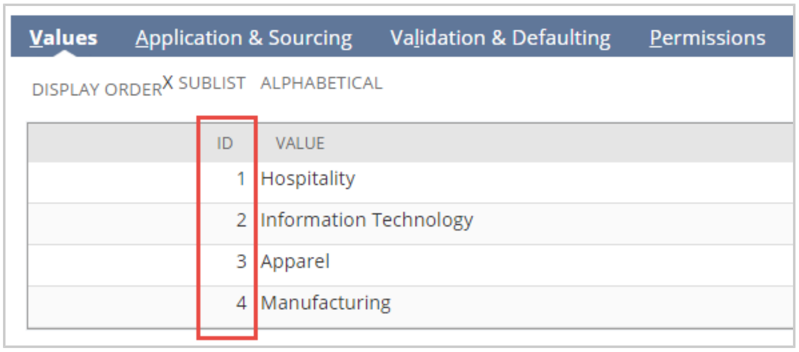
Typically, each custom segment includes a list of possible values. To set a value for a segment in web services, at least one value must already be defined for the segment.

You can select a custom segment value by specifying the value's internal ID. Alternatively, if the value was created through web services, SuiteScript, or CSV import, the value may have an external ID. In this case, you can also use the external ID.

If you choose to use the internal ID, note that you can find it by using getSelectValue, or by looking in the UI. The following procedure explains how to find the value in the UI.

To find a custom segment value's internal ID in the UI:

1. Go to Customization > Lists, Records, & Fields > Custom Segments.
2. In the list, locate the appropriate segment and click the corresponding **View** link.
3. Scroll down to the Values sublist to see a list of all possible values. This list includes a column labeled ID, which identifies each value's internal ID.



ID	VALUE
1	Hospitality
2	Information Technology
3	Apparel
4	Manufacturing

Example: Setting a List/Record Segment Value

The following example shows how to set a value for a custom segment of type List/Record. Specifically, this example shows how to add an inventory item record and set a value for the segment as part of the add request. This example assumes that the segment has already been created with a series of possible values. This example also assumes that the segment has already been applied to the inventory item record type.

C#

```
private void addInventoryItem()
{

   // If this is a OneWorld account, define a
   // subsidiary.

   RecordRef mySubsidiary = new RecordRef();
   mySubsidiary.internalId = "1";
   RecordRef[] subsidiaries = new RecordRef[1];
   subsidiaries[0] = mySubsidiary;


   // Define a tax schedule.

   RecordRef myTaxSchedule = new RecordRef();
   myTaxSchedule.internalId = "2";


   // Define values for the item record's body fields.
```

```
    InventoryItem myItem = new InventoryItem();
    myItem.externalId = "9063BA1";
    myItem.displayName = "Cashmere Mittens";
    myItem.itemId = "Cashmere Mittens";
    myItem.taxSchedule = myTaxSchedule;
    myItem.subsidiaryList = subsidiaries;


    // Define an object to represent the segment field. Because the segment
    // if of type List/Record, you must use SelectCustomFieldRef.


    SelectCustomFieldRef selectCustomFieldRef = new SelectCustomFieldRef();


    // Define an object to represent the segment's value.

    ListOrRecordRef custSelectValue = new ListOrRecordRef();
    custSelectValue.internalId = "1";


    // Set the value of the custom field object.

    selectCustomFieldRef.value = custSelectValue;


    // Identify the specific segment for which you are setting a value.
    // Use the script ID of the segment field as it appears on
    // an instance of the inventory item record.

    selectCustomFieldRef.scriptId = "custitem_cseg_profitcenter";


    // Define an array of CustomFieldRefs that includes the selectCustomFieldRef you
    // just created. (Recall that CustomFieldRef is an abstract class that
    // represents both SelectCustomFieldRef and MultiSelectCustomFieldRef.)
    // Set the item's customFieldList as being equal to this array.

    CustomFieldRef[] customFieldRefArray = new CustomFieldRef[1];
    customFieldRefArray[0] = selectCustomFieldRef;
    myItem.customFieldList = customFieldRefArray;


    // Add the item.

    _service.add(myItem);

}
```

SOAP Request

```
<soap:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
```

```
      <record externalId="9063BA1" xsi:type="q1:InventoryItem" xmlns:q1="urn:accounting_2017_1.
lists.webservices.netsuite.com">
         <q1:taxSchedule internalId="2"/>
         <q1:itemId>Cashmere Mittens</q1:itemId>
         <q1:displayName>Cashmere Mittens</q1:displayName>
         <q1:subsidiaryList>
            <recordRef internalId="1" xmlns="urn:core_2017_1.platform.webservices.netsuite.com"
/>
         </q1:subsidiaryList>
         <q1:customFieldList>
          <customField scriptId="custitem_cseg_profitcenter" xsi:type="SelectCustomFieldRef" xm
lns="urn:core_2017_1.platform.webservices.netsuite.com">
               <value internalId="1"/>
            </customField>
         </q1:customFieldList>
      </record>
   </add>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
   <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <writeResponse>
         <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.web
services.netsuite.com"/>
         <baseRef xsi:type="platformCore:RecordRef" type="inventoryItem" externalId="9063BA1" i
nternalId="346" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
 </addResponse>
</soapenv:Body>
```

Example: Setting Values for Multiple Segments

The following example shows how to set values for two custom segments on the same record instance. Specifically, this example shows how to add a customer record and set values for the segments as part of the add request. One segment is of type List/Record and the other is of type Multiple Select. This example assumes that the segment has already been created with a series of possible values. This example also assumes that the segment has already been applied to the customer record type.

C#

```
private void addCustomer()
{

   // If this is a OneWorld account, define a subsidiary.

   RecordRef mySubsidiary = new RecordRef();
   mySubsidiary.internalId = "1";


   // Define values for the body fields of the customer record.

   Customer myCustomer = new Customer();
   myCustomer.externalId = "5033BA1";
   myCustomer.companyName= "Acme Inc.";
   myCustomer.subsidiary = mySubsidiary;


   // Define a MultiSelectCustomFieldRef object to represent the multiple select segment field.


   MultiSelectCustomFieldRef multiSelectCustomFieldRef = new MultiSelectCustomFieldRef();


   // Create an array of two values. Use the array to identify two of the custom segment’s avai
lable values.

   ListOrRecordRef[] multiSelectValueArray = new ListOrRecordRef[2];
   multiSelectValueArray[0] = new ListOrRecordRef();
   multiSelectValueArray[0].internalId = "1";
   multiSelectValueArray[1] = new ListOrRecordRef();
   multiSelectValueArray[1].internalId = "4";

   multiSelectCustomFieldRef.value = multiSelectValueArray;

   multiSelectCustomFieldRef.scriptId = "custentity_cseg_upsellareas";


   // Define a SelectCustomFieldRef object to represent the list/record segment field.

   SelectCustomFieldRef selectCustomFieldRef = new SelectCustomFieldRef();

   ListOrRecordRef custSelectValue = new ListOrRecordRef();
   custSelectValue.internalId = "1";

   selectCustomFieldRef.value = custSelectValue;
   selectCustomFieldRef.scriptId = "custentity_cseg_ranking";


   // Define the record’s custom field list.

   CustomFieldRef[] customFieldRefArray = new CustomFieldRef[2];
   customFieldRefArray[0] = multiSelectCustomFieldRef;
   customFieldRefArray[1] = selectCustomFieldRef;
   myCustomer.customFieldList = customFieldRefArray;


   // Add the customer.

   _service.add(myCustomer);
}
```

SOAP Request

```
<soap:Body>
```

```
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record externalId="5033BA1" xsi:type="q1:Customer" xmlns:q1="urn:relationships_2017_1.li
sts.webservices.netsuite.com">
         <q1:companyName>Acme Inc.</q1:companyName>
         <q1:subsidiary internalId="1"/>
         <q1:customFieldList>
            <customField scriptId="custentity_cseg_upsellareas" xsi:type="MultiSelectCustomFiel
dRef" xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
               <value internalId="1"/>
               <value internalId="4"/>
          </customField>
            <customField scriptId="custentity_cseg_ranking" xsi:type="SelectCustomFieldRef" xml
ns="urn:core_2017_1.platform.webservices.netsuite.com">
               <value internalId="1"/>
          </customField>
         </q1:customFieldList>
    </record>
   </add>
</soap:Body>
```

SOAP Response

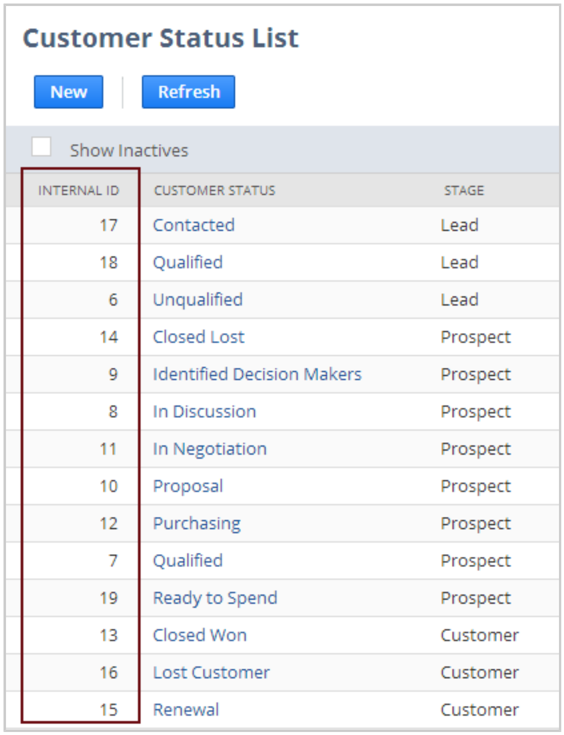
```
<soapenv:Body>
   <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <writeResponse>
         <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.web
services.netsuite.com"/>
            <baseRef xsi:type="platformCore:RecordRef" type="customer" externalId="5033BA1" int
ernalId="4352" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

User Defined Lists

In SuiteTalk, many fields require internal ID values that correspond to an item in a user-defined list. To locate the internal ID of a specific value in a user-defined list, you must have the **Show Internal IDs** preference enabled in your account. For details, see Enabling the Show Internal IDs Preference.

After enabling the **Show Internal IDs** preference, you can then navigate to the appropriate list within NetSuite. The internal ID values (also referred to as nsKeys) are displayed for each list item.

For example, the Status (*entityStatus*) field on the customer record takes an internal ID value from the Customer Status list, which is a user-defined list located at Setup > Sales > Customer Statuses. If you navigate to this page in NetSuite, you can see the internal ID values for each item in the column called **Internal ID** (see figure).



INTERNAL ID	CUSTOMER STATUS	STAGE
17	Contacted	Lead
18	Qualified	Lead
6	Unqualified	Lead
14	Closed Lost	Prospect
9	Identified Decision Makers	Prospect
8	In Discussion	Prospect
11	In Negotiation	Prospect
10	Proposal	Prospect
12	Purchasing	Prospect
7	Qualified	Prospect
19	Ready to Spend	Prospect
13	Closed Won	Customer
16	Lost Customer	Customer
15	Renewal	Customer

Note: Some user-defined lists can also be edited through web services to modify the values for the list. For details, see the help topic Other Lists.

Enumerated Lists

If values being returned for WSEnum and Enum fields in a get or search operation do NOT match the values enumerated in the schema, the following warning is returned:

```
Code = Invalid_data
Message = "Error in record number <id>: Invalid field value <field>.
Please refer to the XSD for enumerated list of valid field values."
```

In these cases, the existing data is corrupt and must be corrected before it can be retrieved using web services.

Field Lengths

Most string fields have a maximum character limit. If the limit is exceeded, an error is returned indicating which field exceeded the limit and the maximum number of characters allowed for that field.

Field Level Errors

If a required field is missing for a specific record within a request, an error is generated. The associated record is not processed and an appropriate error status code is returned.

Only the records without errors are processed. If multiple records are submitted within the same request, records without errors are processed; records with errors are not processed.

Required Fields

Required fields in the UI do not necessarily correspond to required fields in web services. This is because there can be standard defaults that are applied if the field is not populated. For example, in the CalendarEvent record type, the *eventAccess* field is required in the UI. However, it is optional in web services because it has a default value of Public.

Important: The **Required** column in the SuiteTalk Schema Browser lists T or F (for true or false) to specify if a field is required by the form in the user interface. Refer to the **Cardinality** column to see whether the XSD imposes a requirement for the field. For information on using the SuiteTalk Schema Browser, see the help topic SuiteTalk Schema Browser.

Fields and Operations

For simplicity, the SuiteTalk WSDL is designed with all available fields listed for each record in each corresponding XSD. There is no differentiation as to what field is available for each individual operation. For example, add operations take a separate set of field values than the corresponding update operation for some records. If your web services request includes a field value for a field that is unavailable, an error is thrown for that submission.

If you are using web services for data migration where there may be fields that need to be populated that are unavailable during an add operation, you should perform two consecutive requests. Submit an initial add or addList request, with values for all fields available for an add operation, followed by an update or updateList request, with values for the fields available only during an update operation.

Default Field Values

The system provides default values only for fields that are not required. When applying a default value, the system first tries to use a specified value. If none is given, the system uses the default. If no default is given, the system enters null.

User-defined defaults through the UI do not apply to web services.

Hidden Fields

The SuiteTalk API includes fields that are not visible in the NetSuite UI. In some cases, these fields have been hidden through SuiteBuilder point-and-click customization. In other cases, hidden fields are hidden by NetSuite. These types of hidden fields are used primarily for storing system-generated information such as dates.

For example, every SuiteTalk-support record contains either a *dateCreated* or *createdDate* field. These fields are not writeable. By default, these fields are populated with a system-generated timestamp. Note that system-generated dates and times are preserved for audit trail purposes.

Note: The Use Defaults Web service preference does not affect the behavior of hidden fields. Even if the Use Defaults preference is not enabled, hidden fields are populated when a value is not supplied in the request. Also, for audit purposes, the true system date is always recorded for each record and cannot be overridden.

Important: Calls to get or search (when the full record is returned) return the record in **edit** mode.

Forms in Web Services

If you are building a generic SuiteTalk application, to ensure that your applications are account independent, NetSuite recommends that you create **custom "Web-services-only" forms**. Use custom forms for web services requests by specifying the form in the *customForm* element of a record. This will alleviate problems associated with customer-specific customizations to forms used in the UI that may break your web services application code.

Note: If you are unfamiliar with NetSuite custom forms, see the help topic Custom Forms. For information on creating custom forms that you can designate as "web-services-specific," see the help topic Creating Custom Entry and Transaction Forms.

For all record types that support custom forms, if a form is specified in the web services request, that form will be used for validation. Note that it is generally best practice to specify the custom form in all web services update operations. Also be aware that no record type is needed when specifying a custom form. You only need to specify the internal ID of the custom form.

To get the internal ID of a custom form, in the UI, go to Setup > Customization > [*form type*], where form type is either an entry or transaction form. The internal ID appears in the Internal ID column if the Show Internal IDs preference is enabled. (For steps on enabling this preference, see Enabling the Show Internal IDs Preference.)

To specify a form in web services, see the following C# snippet, which shows how to associate a particular form with a Customer record.

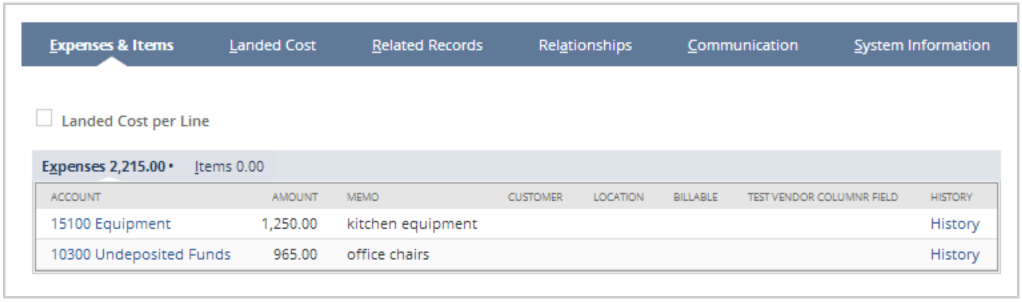
```
// create a customer object

Customer customer = new Customer();

customer.internalId = "555"

// create a custom form object

RecordRef customFormRef = new RecordRef();

// set the internal ID of the custom form. Get the internal ID from
the UI

customFormRef.internalId = "-100";

// set the customForm field on the Customer record to reference the
form

customer.customForm = customFormRef;
```

If a form is NOT specified, then the default preferred form for that record is used. If a custom form is saved with a record in the UI, that form is not used in the web services request unless it is also the default form for that record type or is explicitly set as the form in the web services request.

Sublists in Web Services

Most records in NetSuite include sublists, which generally contain a list of references to other records. This figure shows the Expenses sublist on the check record. Notice that this sublist includes two lines in the Expenses sublist.

Note: For general information on sublists in NetSuite, see the help topic What is a Sublist?



ACCOUNT	AMOUNT	MEMO	CUSTOMER	LOCATION	BILLABLE	TEST VENDOR COLUMNR FIELD	HISTORY
15100 Equipment	1,250.00	kitchen equipment					History
10300 Undeposited Funds	965.00	office chairs					History

In web services, sublists are represented in a record's schema definition by its **List** elements.

Note: Except for the customFieldList element (which contains a record's custom fields), all other elements that end in **List** represent a record's sublists.

The following is a portion of the schema for the Check record. (Although the XSD shown below describes the Items sublist on the check record, the same pattern for defining sublists applies to all sublists on all records.)

Notice that the sublists on the check record include the Expenses sublist, the Items sublist, and the Landed Costs sublist (not shown in the figure above because the Landed Costs feature is not enabled in the account).

```
 <complexType name="Check">
        <complexContent>
            <extension base="platformCore:Record">
                <sequence>
                    <element name="createdDate" type="xsd:dateTime" minOccurs="0"/>
                    <element name="address" type="xsd:string" minOccurs="0"/>
                    <element name="subsidiary" type="platformCore:RecordRef" minOccurs="0"/>
         ........
                    <element name=" expenseList " type="tranBank:CheckExpenseList" minOccurs="0
"/>
                    <element name=" itemList " type="tranBank:CheckItemList" minOccurs="0"/>
                    <element name=" landedCostsList " type="tranBank:CheckLandedCostList" minOc
curs="0"/>
                    <element name="billPay" type="xsd:boolean" minOccurs="0"/>
                    <element name="customFieldList" type="platformCore:CustomFieldList" minOccu
rs="0"/>
                </sequence>
              </extension>
        </complexContent>
```

```
    </complexType>
```

The **itemList** element (listed above) is of type **CheckItemList**, which is defined in the **CheckItem** complex type (defined in transactions.bank.xsd). The **CheckItem** type (shown below) includes all the properties that can be set for the Item sublist on the check record. Also notice the replaceAll attribute in CheckItemList. You will set the value of replaceAll to true or false depending on whether you are working with keyed sublists or non-keyed sublists. See Updating Sublists in Web Services for details.

```
<complexType name=" CheckItem ">
        <sequence>
            <element name="item" type="platformCore:RecordRef" minOccurs="0"/>
            <element name="vendorName" type="xsd:string" minOccurs="0"/>
      ....

            <element name="line" type="xsd:long" minOccurs="0"/>
            <element name="location" type="platformCore:RecordRef" minOccurs="0"/>
            <element name="isBillable" type="xsd:boolean" minOccurs="0"/>
            <element name="customFieldList" type="platformCore:CustomFieldList" minOccurs="0"/>

        </sequence>
    </complexType>
    <complexType name=" CheckItemList ">
        <sequence>
            <element name="item" type="tranBank:CheckItem" minOccurs="0" maxOccurs="unbounded"/
>
        </sequence>
        <attribute name=" replaceAll " type="xsd:boolean" default="true"/>
    </complexType>
```

Updating Sublists in Web Services

When working with sublists, the approach for updating the lines in a sublist will vary depending on whether you are working with Keyed Sublists or Non-Keyed Sublists.

Keyed Sublists

Keyed sublists have an indexing line element or internal ID that can be used as a key. Keyed sublists allow you to set the **replaceAll** attribute to FALSE to update only the lines you are submitting in your SOAP request. When working with keyed sublists, the value of the replaceAll attribute has the following effects:

- replaceAll = TRUE: The existing sublist is **replaced** with the sublist submitted in the web services request. The newly submitted sublist should include all values.
  - Lines that do not match the newly submitted lines are removed.
  - Currently existing lines that match lines in the new sublist submission are updated.
  - Newly submitted lines with no matches are added.

  The default value for the replaceAll attribute is TRUE.
- replaceAll = FALSE: Lines in the existing sublist are **selectively updated** with lines in the sublist submitted in the web services request. The newly submitted sublist need only include values to be added and updated.
  - Lines that do not match the newly submitted lines are preserved.
  - Currently existing lines that match lines in the new sublist submission are updated.

- Newly submitted lines with no matches are added.

The following table lists sublists that are categorized as keyed sublists in web services. Sublists that do not appear in this table are considered Non-Keyed Sublists. The sublist key listed for each keyed sublist below is used to link the sublist data with the appropriate record.

Important: PricingMatrixList and demandPlanDetailList are matrix sublists. Updates to matrix sublists may require additional handling related to the replaceAll attribute. See Matrix Sublists and replaceAll.

Sublist	Sublist Key	Appears on Record(s)
accruedTimeList	payrollitem	Employee
addressbookList	internalId	Contact, Customer, Employee, Lead, Partner, Project, Prospect, Vendor **Note:** When you work with the addressbookList on customer records, be sure to use **internalId** as the key. Do not use **label**.
applyList	doc	Customer Payment, Credit Memo, Customer Refund, Customer Deposit, Deposit Application, Vendor Credit, Vendor Payment
assigneeList	resource	Project Task
attendeeList	attendee	Calendar Event
binNumberList	binNumber	Assembly Item (BOM Item), Inventory Item, Lot Numbered Assemby Item, Lot Numbered Inventory Item, Serialized Assembly Item, Serialized Inventory Item
campaignDirectMailList	internalId	Campaign
campaignEmailList	internalId	Campaign
campaignEventList	internalId	Campaign
companyContribution List	payrollitem	Employee
componentList	item	Assembly Unbuild, Assembly Build
contactList	company	Customer, Partner, Vendor
creditList	doc	Customer Payment, Vendor Payment
creditCardsList	internalId	Customer, Job
currencyList	currency	Customer (when Multi-Currency Customer enabled)
deductionList	payrollitem	Employee
depositList	doc	Customer Payment, Customer Refund
deptAccessList	dept	Custom Entity Field, CRM Custom Field, Other Custom Field, Item Custom Field, Transaction Body Custom Field, Transaction Column Custom Field, Item Option Custom Field,

Sublist	Sublist Key	Appears on Record(s)
		Custom Record Custom Field, Item Number Custom Field
dimensionList	sequence	Fair Value Price
directDepositList	internalid	Employee
earningList	payrollitem	Employee
expCostList	doc	Invoice, Cash Sale
expenseList	orderLine or line	Check, Expense Report, Item Receipt, Purchase Order, Purchase Requisition, Vendor Bill, Vendor Credit, Vendor Return Authorization
itemList	orderLine or line	Bin Putaway Worksheet, Bin Transfer, Cash Sale, Check, Estimate, Inventory Transfer, Invoice, Item Fulfillment, Item Receipt, Credit Memo, Return Authorization, Cash Refund, Opportunity, Purchase Order, Purchase Requisition, Sales Order, Vendor Bill, Vendor Credit, Vendor Return Authorization, Work Order Notes: On Item sublists, the value **End of Group** is not returned in web services. This behavior mimics the behavior of the UI. For cash sale records, the orderLine field establishes the relationship with an existing sales order, if any. You must set a value for orderLine to populate the createdFrom field. For item receipt records, orderLine is used for transforms, because it implies a link between the previous transaction and the current one. For example, to add an item receipt from a purchase order, the purchase order lines would be "orderLines", because the receipt has not been saved. After the item receipt is saved, lines should be accessed via the "line" field. For opportunity records, line is used.
itemCostList	doc	Invoice, Cash Sale
itemsList	item	Promotion Code, Item Option Custom Field, Item Number Custom Field.
lineList	line	Custom Transaction
locationsList	locationId	Inventory Item, Serialized Inventory Item, Lot Numbered Inventory Item
milestoneList	milestoneId	Billing Schedule **Note:** This sublist is available only when scheduleType is set to Fixed Bid Milestone.
nexusesTaxList	nexus	Tax Type

Sublist	Sublist Key	Appears on Record(s)
partnersList	partner	Estimate, Cash Sale, Invoice, Sales Order, Opportunity, Credit Memo, Return Authorization, Cash Refund, Customer
partnersList	partner	Promotion Code
paymentList	The internal ID of the record being referenced in the sublist line.	Deposit
periodDemandPlanList	startDate	Item Demand Plan **Note:** This sublist includes additional handling for replaceAll. See Matrix Sublists and replaceAll.
predecessorList	task	Project Task
pricingMatrixList	Varies according to enabled features. See the help topic Pricing Matrix Keys.	Item **Note:** This sublist includes additional handling for replaceAll. See Matrix Sublists and replaceAll.
ratesList	subsidiary	Expense Category **Note:** Sublist available only to OneWorld accounts.
recurrence	recurrenceId	Billing Schedule **Note:** This sublist is available only when scheduleType is set to Standard and frequency is set to Custom.
resourceList	resource	Calendar Event
roleAccessList	role	Entity Custom Field, CRM Custom Field, Other Custom Field, Item Custom Field, Transaction Body Custom Field, Transaction Column Custom Field, Item Option Custom Field, Custom Record Custom Field, Item Number Custom Field
salesTeam	employee	Customer, Invoice, Cash Sale
timeList	doc	Invoice, Cash Sale

Example

The following sample shows how to add an item to an opportunity record and modify an existing item in that record. To do this, set replaceAll to FALSE, add the new item, and modify the existing item by referencing the desired item in the line property. The following shows that the existing line item is modified to change the quantity.

```
<soapenv:Body>
```

```
<platformMsgs:update xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:s0="urn:sales_2011_2017_1.transactions.webservices.netsuite.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:platformMsgs="urn:messages_2017_1.platform.webservices.netsuite.com">
<platformMsgs:record xsi:type="s0:Opportunity" internalId="1638">
   <s0:itemList replaceAll="false">
      <s0:item>
         <s0:item internalId="380" type="inventoryItem" />
         <s0:quantity>1.0</s0:quantity>
         <s0:amount>20.0</s0:amount>
      </s0:item>
      <s0:item>
         <s0:line>2</s0:line>
         <s0:quantity>10.0</s0:quantity>
      </s0:item>
   </s0:itemList>
</platformMsgs:record>
</platformMsgs:update>
</soapenv:Body>
```

Matrix Sublists and replaceAll

Matrix sublists respect the replaceAll attribute for updates, as other Keyed Sublists do, but in some cases, updates to matrix sublists require additional handling.

Matrix sublists require additional handling for replaceAll because body-level fields slave values into them. For example, setting the pricing schedule for an item dictates values in the pricing matrix. In the UI, when a pricing schedule is selected, the pricing schedule logic is run first, and values specified in the pricing matrix sublist are taken into account to arrive at what the update should do. However, the UI does not have the notion of replaceAll because the user interacting with the form can delete values they do not want to keep in individual fields.

Web services requests cannot delete individual field values in a sublist. The only way to get rid of old values is to set replaceAll to TRUE and send all values in the update request. Some body fields can override values in a matrix sublist, but a web services request can then change some or all of these sublist values, depending on the replaceAll setting.

Review the following cases to get an understanding of how the slaving of values from body fields into a matrix sublist interacts with the replaceAll setting on updates:

- No sublist values from body fields and replaceAll=TRUE

  The web services request does not include any body field values that drive values in the sublist, so the update is handled like those in other keyed sublists where replaceAll is set to TRUE.

  The existing sublist is **replaced** with the sublist submitted in the web services request. The newly submitted sublist should include all values.
- No sublist values from body fields and replaceAll = FALSE

  The web services request does not include any body field values that drive values in the sublist, so the update is handled like those in other keyed sublists where replaceAll is set to FALSE.

  Lines in the existing sublist are **selectively updated** with lines in the sublist submitted in the web services request. The newly submitted sublist need only include values to be added and updated.
- Sublist values from body fields and replaceAll=TRUE

  The web services request sets body field value(s) that drive values in the sublist, so specialized handling is required.

First, all sublist values are removed. Next, sublist values driven by body field values are set. Then, the sublist is updated with all lines specified in the newly submitted sublist. The newly submitted sublist lines thus override the values driven by body fields.

Important: When sublist values from a web services request override values driven by body fields, errors may occur. If you do not want these overrides, you must set replaceAll to FALSE. For example, if you are setting a quantity pricing schedule, you should set ReplaceAll to FALSE.

- Sublist values from bodyfields and replaceAll=FALSE

  The web services request sets body field value(s) that drive values in the sublist, so specialized handling is required.

  First, sublist values driven by body field values are set. Then, the sublist is updated with any updated or added lines from the newly submitted sublist.

Note: By default, replaceAll is set to TRUE for matrix sublists, like other sublists.

Currently Supported Matrix Sublists

The pricingMatrixList on Items and periodDemandPlanList on Item Demand Plans are currently supported matrix sublists. For details about these sublists, see the help topics Pricing Matrix List and Item Demand Plan.

Important: In endpoints prior to 2012.1, updates to pricingMatrixList ignore the replaceAll attribute. For the 2012.1 and later endpoints, replaceAll is respected for this sublist.

Non-Keyed Sublists

Non-keyed sublists have no indexing line element or internal ID that can be used as a key. Each line in a non-keyed sublist is recorded in a *flat* list.

In the UI, non-keyed sublists and keyed sublists may look similar; however, technically, they are quite different. Because non-keyed sublists contain no referencing keys (or handles), you cannot update a specific line in a non-keyed sublist. Instead, you must interact with the **sublist as a whole.** In non-keyed sublists, the **replaceAll** attribute is ignored and behaves as if it were set to TRUE for all requests. Consequently, an update operation is similar to the add operation with respect to non-keyed sublists.

In the context of web services, all sublists not listed as Keyed Sublists are considered to be non-keyed sublists. The following bullets outline the behaviors of non-keyed sublists.

- To update a single line in the sublist, retrieve the entire sublist, change a single line as desired, and then resubmit the entire sublist.
- To replace the sublist with a subset of lines, retrieve the entire sublist, and then resubmit the subset of lines. The original sublist is purged and replaced by the new sublist containing the subset of lines.
- To delete the sublist, submit an empty sublist. See Deleting All Lines on a Sublist for more information.

Sublist Line Numbers

The index number for each sublist line is assigned by the system. Web services developers cannot change these values. When lines are added or removed from a sublist (using either the UI or web services), the system does not re-index the line numbers. Therefore, it is best practice to perform a get operation on the sublist data before trying to update individual lines, otherwise you may be updating the wrong line in the sublist.

For example, if you perform a get operation on an Items sublist on a transaction record, the index values for the first three line items might be:

<ns3:line>1</ns3:line>
<ns3:line>8</ns3:line>
<ns3:line>9</ns3:line>

Deleting All Lines on a Sublist

This sample shows how to delete all addresses from the Address sublist on a customer record (CustomerAddressbookList).

Java

```
Customer update = new Customer();
update.setInternalId(c.getInternalId());
cabl = new CustomerAddressbookList();
cabl.setAddressbook(new CustomerAddressbook[0]);
cabl.setReplaceAll(true);
update.setAddressbookList(cabl);
```

SOAP

```
<update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record internalId="724" xsi:type="ns1:Customer" xmlns:ns1="urn:relationships_2017_
1.lists.webservices.netsuite.com">
               <ns1:addressbookList replaceAll="true" xsi:type="ns1:CustomerAddressbookList"/>
            </record>
         </update>
```

The next sample shows how to delete all line items from the Item sublist (OpportunityItemList) on an opportunity record.

Java

```
Opportunity update = new Opportunity();
update.setInternalId(opp.getInternalId());
update.setItemList(new OpportunityItemList(new OpportunityItem[0], true));
```

SOAP

```
<update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record internalId="6147" xsi:type="ns1:Opportunity" xmlns:ns1="urn:sales_2017_1.tr
ansactions.webservices.netsuite.com">
               <ns1:itemList replaceAll="true" xsi:type="ns1:OpportunityItemList"/>
```

```
        </record>
    </update>
```

Searching a Sublist

Keyed sublists have a **line** field (typically on transaction sublists). Individual lines in the list can be searched for by referencing the line value.

Also note that when searching records, by default sublists are NOT returned in the result set, which increases the search response time. However, if you want to return all sublist data in your search, set the SearchPreferences. *bodyFieldsOnly* preferences to FALSE in your search request.

Web Services Processing

This section describes synchronous and asynchronous processing. Also provided are steps for monitoring your web services requests. Refer to the following topics:

- Synchronous Versus Asynchronous Request Processing
- Asynchronous Request Processing
- Synchronous Request Processing

Synchronous Versus Asynchronous Request Processing

Web services requests can be processed synchronously or asynchronously.

With synchronous requests, your client application sends a request to NetSuite, and the client waits until the request is processed and a response is returned. That is, the application does not proceed with other work until receiving the response.

With asynchronous requests, your client application sends a request to SuiteTalk, where it is placed in a processing queue and handled asynchronously with other requests. Your client application does not wait for a response but goes on to other work. After a job is submitted, a job Id is returned in the web services response. Your client application can then check on the status and result of the request by referencing the job Id.

Asynchronous processing may be advantageous in the following situations:

- If you expect your connection to NetSuite to be slow or unstable.
- If your job is large, and its processing can be postponed until off-peak hours.

Be aware that asynchronous responses may not be returned immediately. Before committing to using asynchronous operations, you should consider this factor and decide whether it fits in with your business logic.

Asynchronous Request Processing

This section describes asynchronous request processing in more detail.

Available Operations

Asynchronous equivalents are available for several synchronous operations, as shown in the table below.

Asynchronous Operation	Synchronous Equivalent
asyncAddList	addList
asyncUpdateList	updateList

Asynchronous Operation	Synchronous Equivalent
asyncUpsertList	upsertList
asyncDeleteList	deleteList
asyncGetList	getList
asyncSearch	search
asyncInitializeList	initializeList

Understanding Asynchronous Requests and Responses

When submitting an asynchronous request, the request is similar to the equivalent synchronous call. The following code illustrates an asynchronous request for a list of customer records.

```
<soap:Body>
   <platformMsgs:asyncGetList>
      <platformMsgs:baseRef internalId="87" type="customer" xsi:type="platformCore:RecordRef"><
/platformMsgs:baseRef>
      <platformMsgs:baseRef internalId="176" type="customer" xsi:type="platformCore:RecordRef">
</platformMsgs:baseRef>
   </platformMsgs:asyncGetList>
</soap:Body>
```

The response, however differs in that only information about the request is returned. This information includes a unique identifier for the request, which is enclosed in the jobId tags.

```
<soapenv:Body>
   <asyncGetListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <asyncStatusResult xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
         <jobId>ASYNCWEBSERVICES_563214_053120061943428686160042948_4bee0685</jobId>
         <status>pending</status>
         <percentCompleted>0.0</percentCompleted>
         <estRemainingDuration>0.0</estRemainingDuration>
      </asyncStatusResult>
   </asyncGetListResponse>
</soapenv:Body>
```

Later, you can use the checkAsyncStatus or getAsyncResult operations to track the asynchronous request. To identify the specific request about which you need data, refer to it by its job Id.

Usage Notes

Be aware of the following characteristics of asynchronous processing:

- All available jobs are processed contiguously. That is, processing priority is based on when the job is submitted, and jobs cannot pass one another in the queue.
- The system does not enforce waiting periods for jobs. That is, as soon as soon as a job is completed, the next job from the queue is immediately selected and the processing starts.
- Asynchronous request job Ids are valid for 30 days. After 30 days, you can no longer look up the request.

Checking the Status of an Asynchronous Job

To check the status of an asynchronous web services job, you use the checkAsyncStatus operation. When you use this operation, you reference a job Id. In response, the system returns the job's status, among other details.

Specific status values that can be returned include the following:

- failed
- finishedWithErrors — indicates that at least one record was processed, but at least one failed.
- pending
- processing
- finished

If the status is failed, finishedWithErrors, or finished, you can in most cases use the getAsyncResult operation to obtain more details. These details may include the record results, or error and fault messages. The getAsyncResult operation is described in Checking for Detailed Results of an Asynchronous Job.

checkAsyncStatus also returns details about the percentage of the job that is complete, and the estimated time remaining.

checkAsyncStatus Request and Response

The following is a typical checkAsyncStatus request.

```
<soap:Body>
   <platformMsgs:checkAsyncStatus>
      <platformMsgs:jobId>ASYNCWEBSERVICES_563214_053120061943428686160042948_4bee0685
      </platformMsgs:jobId>
   </platformMsgs:checkAsyncStatus>
</soap:Body>=
```

The following is a typical checkAsyncStatus response.

```
<soapenv:Body>
<checkAsyncStatusResponse
   xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <asyncStatusResult xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
      <jobId>ASYNCWEBSERVICES_563214_053120061943428686160042948_4bee0685</jobId>
      <status>pending</status>
      <percentCompleted>0.0</percentCompleted>
      <estRemainingDuration>0.0</estRemainingDuration>
   </asyncStatusResult>
</checkAsyncStatusResponse>
</soapenv:Body>
```

checkAsyncStatus Faults

The checkAsyncStatus operation can throw any of the following faults:

- InvalidSessionFault

- AsyncFault
- UnexpectedErrorFault

For more information on faults, see SOAP Fault Status Codes.

Checking for Detailed Results of an Asynchronous Job

To retrieve details about specific records submitted as part of an asynchronous request, you use the getAsyncResult operation.

getAsyncResult Time Limits

Be aware of the following limits:

- You can use the getAsyncResult operation for a particular job Id up to 20 times within a 30-day time period. If this limit is exceeded, the system returns an INVALID_JOBID user error.
- Async job IDs are purged from the system every 30 days. If you attempt to download the results 31 days after your initial request, the system returns an INVALID_JOBID user error.

getAsyncResult Example

You can use code like the following to retrieve the getAsyncResult response. This example shows how to look for records that failed with a status detail code of “unexpected error” and then take some action with those records.

```
AsyncAddListResult result = (AsyncAddListResult)port.getAsyncResult("job ID", 1);
WriteResponseList responses = result.getWriteResponseList();
for (WriteResponse response : responses.getWriteResponse())
{
   if (!response.getStatus().isIsSuccess())
   {
      if (response.getStatus().getStatusDetail()[0].getCode() == StatusDetailCodeType.UNEXPECTE
D_ERROR)
      // add for resubmit
   }
}
```

Note: To use getAsyncResult with asyncInitializeList, you must be using the 2013.2 endpoint or later. You can monitor asyncInitializeList requests using the UI, as described in Monitoring Asynchronous Jobs from the UI.

getAsyncResult Faults

This operation can throw one of the following faults.

- InvalidSessionFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- AsyncFault

- UnexpectedErrorFault

For more information on faults, see SOAP Fault Status Codes.

Note: You can see some details about asynchronous job results by navigating to Setup > Integration > Web Services Process Status and viewing the Response.xml file for the job. For details, see Monitoring Asynchronous Jobs from the UI.

Monitoring Asynchronous Jobs from the UI

Web services jobs submitted asynchronously can be monitored at Setup > Integration > Web Services Process Status. To access this log, either the full Web Services permission or the View Web Services Logs permission is needed. Both permissions are available on the role record's Permissions subtab under Setup.

On this page, you can view the following information about each job:

- Date — the date the job was created
- Sequence — the order of the job in relation to other asynchronous jobs in process (jobs are processed on a first in, first out basis)
- Job Number — order number of the job in the work queue
- Job Id — the Job Id, which is also returned in the asynch SOAP response, and can be used to programmatically retrieve job status information
- Status — the status of the job: failed, finishedWithErrors, pending, processing, or finished (for more on status, see Checking the Status of an Asynchronous Job)
- Percent Complete — the percentage of job processing that has been completed
- Est. Remaining Time — the estimated amount of time remaining before the job is completed
- Request — the SOAP request associated with the job (see additional details below)
- Response — the SOAP response associated with the job (see additional details below)
- Cancel — you can check the box in this column to cancel a job that has not yet successfully completed

Limits on Requests and Responses

Note the following limitations:

- A response for a job is available only after the job has successfully completed.
- In production environments, the data is accessible for 21 days. In sandbox environments, the data is accessible for seven days.
- Lengthy requests and responses are not saved by the system and therefore are not available for your review. The system imposes the following limits:
  - The system does not save requests that are larger than 100MB.
  - The system does not save responses that are large than 1MB.

Refreshing the Job Status Page

The Job Status page does not automatically refresh. To get the current status of a job, click the Refresh button at the top of the page.

Synchronous Request Processing

In addition to using the responses NetSuite sends for each request, you can monitor synchronous web services processing in the NetSuite user interface. You have the following options:

- Referring to the execution log of the appropriate integration record. For details on working with integration records, see Integration Management.
- Using the Web Services Usage Log
- Creating Integration Reports

See also Handling of Lengthy Requests, which describes limitations that exist for sending responses.

For information on monitoring asynchronous requests, see Monitoring Asynchronous Jobs from the UI.

Using the Web Services Usage Log

Use the Web Services Usage Log to monitor synchronous requests. This page can be accessed at Setup > Integration > Web Services Usage Log.

Only administrators can access the Web Services Usage Log. Assigning the Web Services permission to a role does not provide access to this log. Access is provided to administrators only because access to this log could allow users to see data they do not have permission to view.

The Web Services Usage Log allows users to filter their web services requests by specified date and time periods, record type, and action (operation). The SOAP requests and responses for each job are accessible in the Request and Response columns. In production environments, requests and responses are accessible for **21 days**.

In sandbox environments, requests and responses are accessible for seven days.

Note: In sandbox environments, you can view request and response information in log and search results after seven days. However, links to the original SOAP request and responses are not available.

Results can be filtered down to the minute. The results returned provide second-level granularity.

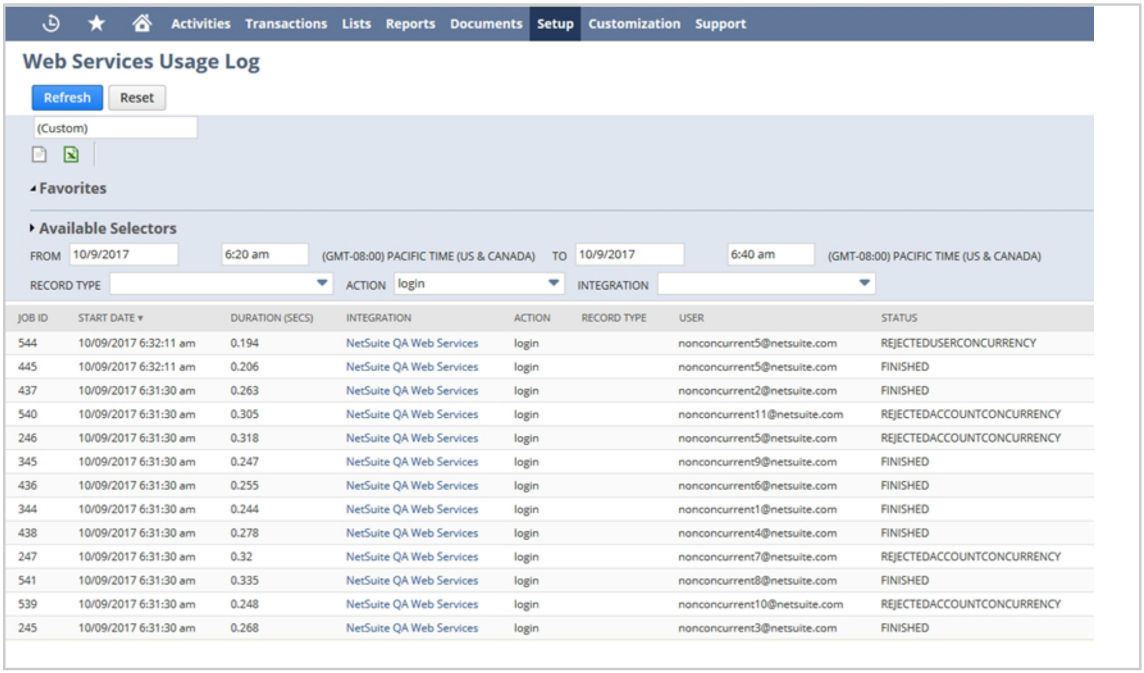
Note that lengthy requests and responses are not saved by the system and therefore are not available for your review. The system imposes the following limits:

- The system does not save requests that are larger than 100MB.
- The system does not save responses that are large than 1MB.

For each request, the log includes details such as the following:

- The request job ID.
- The date and time that the request was made.
- The duration of the request.
- Information about the requests that were rejected due to a violation of the concurrency governance.
- The integration record used for the request.
- The type of action used in the request.
- The record type that was the subject of the request.
- The email address of the user who sent the request.

- The status of the request.
- The number of records included in the request.
- The number of request records that have finished.
- The number of request records that failed.
- The number of records returned in the request's query.
- Links to the SOAP request and response.



Web Services Usage Log

(Custom)

Available Selectors

FROM 10/9/2017 6:20 am (GMT-08:00) PACIFIC TIME (US & CANADA) TO 10/9/2017 6:40 am (GMT-08:00) PACIFIC TIME (US & CANADA)

RECORD TYPE ACTION login INTEGRATION

JOB ID	START DATE ▾	DURATION (SECS)	INTEGRATION	ACTION	RECORD TYPE	USER	STATUS
544	10/09/2017 6:32:11 am	0.194	NetSuite QA Web Services	login		nonconcurrent5@netsuite.com	REJECTEDUSERCONCURRENCY
445	10/09/2017 6:32:11 am	0.206	NetSuite QA Web Services	login		nonconcurrent5@netsuite.com	FINISHED
437	10/09/2017 6:31:30 am	0.263	NetSuite QA Web Services	login		nonconcurrent2@netsuite.com	FINISHED
540	10/09/2017 6:31:30 am	0.305	NetSuite QA Web Services	login		nonconcurrent11@netsuite.com	REJECTEDACCOUNTCONCURRENCY
246	10/09/2017 6:31:30 am	0.318	NetSuite QA Web Services	login		nonconcurrent5@netsuite.com	REJECTEDACCOUNTCONCURRENCY
345	10/09/2017 6:31:30 am	0.247	NetSuite QA Web Services	login		nonconcurrent9@netsuite.com	FINISHED
436	10/09/2017 6:31:30 am	0.255	NetSuite QA Web Services	login		nonconcurrent6@netsuite.com	FINISHED
344	10/09/2017 6:31:30 am	0.244	NetSuite QA Web Services	login		nonconcurrent1@netsuite.com	FINISHED
438	10/09/2017 6:31:30 am	0.278	NetSuite QA Web Services	login		nonconcurrent4@netsuite.com	FINISHED
247	10/09/2017 6:31:30 am	0.32	NetSuite QA Web Services	login		nonconcurrent7@netsuite.com	REJECTEDACCOUNTCONCURRENCY
541	10/09/2017 6:31:30 am	0.335	NetSuite QA Web Services	login		nonconcurrent8@netsuite.com	FINISHED
539	10/09/2017 6:31:30 am	0.248	NetSuite QA Web Services	login		nonconcurrent10@netsuite.com	REJECTEDACCOUNTCONCURRENCY
245	10/09/2017 6:31:30 am	0.268	NetSuite QA Web Services	login		nonconcurrent3@netsuite.com	FINISHED

Note: The Web Services Usage Log does not show asynchronous requests. For details, see Monitoring Asynchronous Jobs from the UI.

Creating Integration Reports

You can monitor web services processing by creating the following types of integration reports:

- Integration and Automation Usage Summary by Job
- Integration and Automation Usage Summary by Record Type

Important: The **Integration** permission is required to view these reports. To enable this permission, administrators must go to Setup > Users/Roles > Manage Roles. Click Customize next to the appropriate role. On the Permissions tab, click Reports. In the Permissions dropdown list, select **Integration** and click Add.

Note: You might receive a timeout error when creating integration reports for accounts with large volumes of web services traffic. Also, although integration reports do offer some filtering capabilities, users must first take the added step of customizing the report. Be aware that you can apply filtering in the Web Services Usage Log without doing customization.

Both of these reports provide details on type of operations used (for example, add, addList, delete), who performed an operation, and the time an operation was performed.

Web Services Integration reports are also useful for administrators who need to diagnose and troubleshoot issues. Note that each report can be customized to display only the desired information. To customize a report, click Customize next to the desired report and then modify the report as desired. For details about how to customize a report, see the help topic Report Customization.

If an integration report is timing out, it is recommended that you customize the report by adding filters so the report returns fewer results. See Filtering Integration Report Data.

Note: These reports do not support reporting by period even when the Report by Period preference is set to All Reports. The Report by Period preference can be configured at Home > Set Preferences, the Analytics subtab.

Integration and Automation Usage Summary by Job

This report can be used for performance statistics. It shows the duration of each web services processing job, the number of records modified in each job, the number of successful versus failed record modifications for each job, and the number of records queried for each job. You can also drill down to job details by clicking the job number.

Any related SOAP files are stored in View links in the reports Request Doc and Response Doc columns. Be aware that NetSuite purges SOAP files every 30 days. In sandbox environments, these files are accessible for seven days.

Note: In sandbox environments, you can view request and response information in log and search results after 7 days. However, links to the original SOAP request and responses are not available.

You can view this report at Reports > Integration > Integration and Automation Usage Summary by Job.

Integration and Automation Usage Summary by Record Type

This report lists the record types modified by web services processing, and the number of records added, updated, deleted, and queried for each record type.

You can view this report at Reports > Integration > Integration and Automation Usage Summary by Record Type.

Filtering Integration Report Data

You can customize integration reports to run faster, by adding filters that limit the returned results. For example, you could limit a report to return only jobs with start dates that fall into a specified date range.

To filter an integration report by start date:

1. Go to Reports > Integration > Integration and Automation Usage Summary by Job > Customize or Reports > Integration > Integration and Automation Usage Summary by Record Type > Customize.
2. Click **Filters**.
3. Under **Search Fields** box, enter **Start Date** and click the **Search** button.

   The **Actual Job Start Date** and **Start Date** fields are listed.
4. Click **Start Date**.

   This field is added to the **Choose Filters** pane.

5. In the **Choose Filters** pane:
    1. Set **Filter** to between.
    2. Set **Date Range** to custom.
    3. In the **From** and **To** fields, enter dates in {mm/dd/yyyy} format.
    4. Click **Done**.
6. Click **Save**.

Handling of Lengthy Requests

Throughout NetSuite, the maximum duration for a single HTTP connection is 15 minutes. However, in some cases a web services request may require more than 15 minutes to process. Because the system cannot send a response after the connection has timed out, the system handles lengthy requests differently from those that complete more quickly. The exact handling varies depending on the operation. For details, see the following:

- Single-Record Requests
- List and Search Requests

Single-Record Requests

If a synchronous single-record operation does not complete within 15 minutes, the operation is allowed to finish. In these cases, no response is sent, but the system updates the Web Services Usage Log with a partial response. However, note that single-record operations almost always complete before the HTTP connection times out.

A single-record operation is one that creates or affects only one record. For example, add, update, and delete are single-record operations.

List and Search Requests

In general, a list or search request that requires longer than 15 minutes to process is terminated by the system. The exact behavior varies depending on the endpoint.

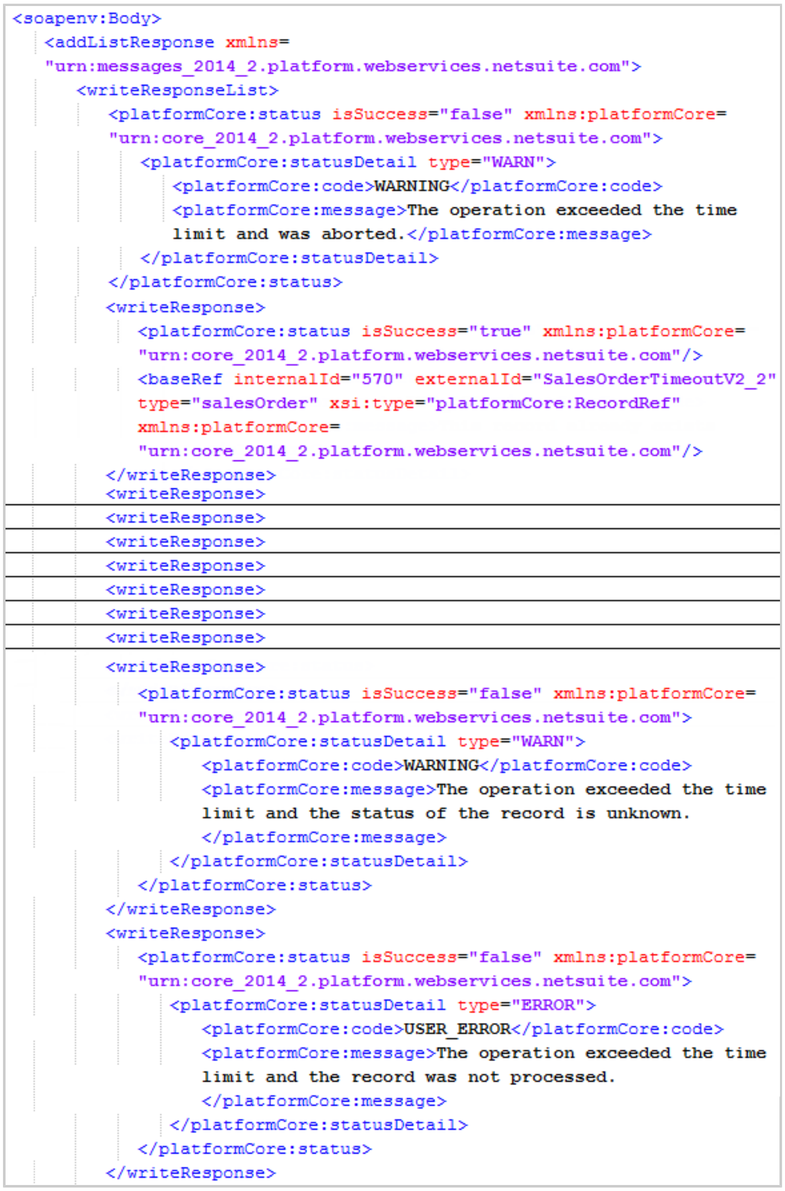
2014.2 Endpoint

When the 2014.2 or a later endpoint is being used, NetSuite sends a response 10 seconds prior to the connection timeout. The operation is terminated after the record currently being processed has finished. The system uses the messaging described in the following table.

Operation Type	Relevant Operations	Messaging
Search	■ search ■ searchMore ■ searchMoreWithId ■ searchNext	The response indicates that the operation did not finish because it timed out.
List	■ addList ■ deleteList ■ getList	The response indicates that the operation did not finish because it timed out. For records that were successfully processed, the response indicates success

Operation Type	Relevant Operations	Messaging
	▪ initializeList (for details, see initialize / initializeList) ▪ updateList ▪ upsertList	for that record, along with other standard details, such as the record's internal ID. For other records, the status indicates one of the following: ▪ The record's status is unknown – This status is used if the processing of the record was still in progress when the response was being created. ▪ The record was not processed. Using this information and the original SOAP request, you can determine which data to resend.

The following screenshot shows a response for an addList operation that was terminated.



```
<soapenv:Body>
   <addListResponse xmlns=
   "urn:messages_2014_2.platform.webservices.netsuite.com">
      <writeResponseList>
         <platformCore:status isSuccess="false" xmlns:platformCore=
         "urn:core_2014_2.platform.webservices.netsuite.com">
            <platformCore:statusDetail type="WARN">
               <platformCore:code>WARNING</platformCore:code>
               <platformCore:message>The operation exceeded the time
               limit and was aborted.</platformCore:message>
            </platformCore:statusDetail>
         </platformCore:status>
         <writeResponse>
            <platformCore:status isSuccess="true" xmlns:platformCore=
            "urn:core_2014_2.platform.webservices.netsuite.com"/>
            <baseRef internalId="570" externalId="SalesOrderTimeoutV2_2"
            type="salesOrder" xsi:type="platformCore:RecordRef"
            xmlns:platformCore=
            "urn:core_2014_2.platform.webservices.netsuite.com"/>
         </writeResponse>
         <writeResponse>
         <writeResponse>
         <writeResponse>
         <writeResponse>
         <writeResponse>
         <writeResponse>
         <writeResponse>
         <writeResponse>
            <platformCore:status isSuccess="false" xmlns:platformCore=
            "urn:core_2014_2.platform.webservices.netsuite.com">
               <platformCore:statusDetail type="WARN">
                  <platformCore:code>WARNING</platformCore:code>
                  <platformCore:message>The operation exceeded the time
                  limit and the status of the record is unknown.
                  </platformCore:message>
               </platformCore:statusDetail>
            </platformCore:status>
         </writeResponse>
         <writeResponse>
            <platformCore:status isSuccess="false" xmlns:platformCore=
            "urn:core_2014_2.platform.webservices.netsuite.com">
               <platformCore:statusDetail type="ERROR">
                  <platformCore:code>USER_ERROR</platformCore:code>
                  <platformCore:message>The operation exceeded the time
                  limit and the record was not processed.
                  </platformCore:message>
               </platformCore:statusDetail>
            </platformCore:status>
         </writeResponse>
```

2014.1 Endpoint and Earlier

If you are using 2014.1 or an earlier endpoint, a lengthy operation is stopped after 15 minutes. The system does not send a response, but it updates the Web Services Usage Log with a partial response, which shows only the records that were processed. For help with the Web Services Usage Log, see Using the Web Services Usage Log.

Web Services Security

NetSuite leverages the latest industry security standards to ensure high levels of security around your business data. All web services requests are controlled by the same security measures used in the NetSuite UI. This includes:

- Authentication for Web Services
- Authorization for Web Services
- Session Management for Web Services
- Encryption for Web Services

Additional security-related topics include:

- Custom Field Security - for information on security applied at the field level and its affect on web services
- PCI Compliance Password Requirements - for information on password requirements associated with the Payment Card Industry (PCI) Data Security Standard

Authentication for Web Services

Authentication is the process of determining the identity of requesters by verifying the credentials they present. Most web services operations require authentication.

The available methods of authentication include the following:

- User Credentials
- Token-Based Authentication
- Inbound Single SignOn (Inbound SSO)
- Outbound Single Sign-on (SuiteSignOn)

Important: Use only one authentication method in one SOAP message. Mixing different authentication types in a single SOAP message returns a SOAP fault.

User Credentials

With this approach, the request must include the credentials associated with a single user: an email address, password, role, and NetSuite account ID. To find your account ID, go to the Web Services Preferences page at Setup > Integration > Integration Management > Web Services Preferences. For a list of NetSuite role internal IDs, see Internal IDs Associated with Roles.

When you authenticate by using user credentials, there are two approaches to how the SOAP request may be structured: you can use the login operation or you can use request-level credentials. For details, see Structuring SOAP Requests to Include Authentication Details.

For requests that use the 2015.2 WSDL or later, application ID is also mandatory, although technically application ID is not considered a part of the user authentication process. An application ID is used to link a request to an integration record that exists in your NetSuite account. To use any integration record for requests that authenticate through user credentials, the record must have the **User Credentials** box checked. For details, see Integration Management.

Token-Based Authentication

With this approach, the request must include a consumer key, a token, and other data. Authentication details must be sent by using request-level credentials, not the login operation. For details on structuring a SOAP request this way, see Request-Level Credentials.

You generate a consumer key and secret when you create an integration record and configure the record to permit token-based authentication (by checking the **Token-based Authentication** box). For details, see Integration Management.

Important: Token-based authentication is supported for use only with the 2015.2 endpoint and later.

Inbound Single SignOn (Inbound SSO)

This approach is used to provide authentication to NetSuite on behalf of an individual who has already authenticated to an external application.

With inbound SSO, a mapping is created between a user’s NetSuite account and the user’s external account. The mapping can be created by using the mapSso operation, or it can be created manually in the UI. After the mapping is created, the user can be automatically logged in to NetSuite by using the ssoLogin operation, which is described in ssoLogin. The ssoLogin operation requires values representing the partner, the user’s ID in the partner system, and other details.

Note that the ssoLogin operation functions in much the same way as the login operation. For more details, see Login and ssoLogin Operations.

For mapSso and ssoLogin requests that use the 2015.2 WSDL or later, application ID is mandatory, although technically application ID is not considered a part of the authentication process. An application ID is used to link a request to an integration record that exists in your NetSuite account. To use an integration record for requests that authenticate through inbound SSO, the record must have the **User Credentials** box checked. For details, see Integration Management.

Outbound Single Sign-on (SuiteSignOn)

This approach allows an external application to perform callbacks to NetSuite after the SuiteSignOn handshake is complete and there is a live SuiteSignOn token. See the SuiteSignOn Sequence Diagram and Connection Details for details.

For more information about SuiteSignOn, see the help topic Outbound Single Sign-on (SuiteSignOn).

Structuring SOAP Requests to Include Authentication Details

If your chosen authentication method is **user credentials**, the SuiteTalk platform supports two approaches for structuring your SOAP requests. These approaches are described in the following table.

If your authentication method is **inbound SSO**, you must use the first style of authentication listed below. For **token-based authentication**, you must use request-level credentials.

Note: For more information about token-based authentication, see Token-Based Authentication Details.

Name	Valid with	Notes
Login/ssoLogin operation	■ Authentication through user credentials ■ Inbound SSO	This approach initiates a session that is governed by timeout and user limits. For details, see Login and ssoLogin Operations.
Request-level credentials	■ Authentication through user credentials ■ Token-based authentication	You can send credentials in the SOAP header of every request. This approach initiates a stateless form of web services communication in which the login operation is not invoked and certain timeouts do not apply. For details, see Request-Level Credentials. For information about using the TokenPassport complex type with token-based authentication, see TokenPassport Complex Type and Updating an Integration to Send Token-Based Authentication Details.

Note: If you are using outbound SSO, the information in this topic does not apply. For details on outbound SSO, see the help topic Outbound Single Sign-on (SuiteSignOn).

Login and ssoLogin Operations

When you use the login operation, typically you would create a login class that includes a Passport object. Then you would call the login operation, passing the passport object as the sole argument. This is how the process works for user credentials.

With inbound SSO, the process is the same: you would create an ssoPassport object. Then you call the ssoLogin operation, passing the ssoPassport object as the sole argument.

The following C# snippet shows how to create a login class for a request that uses user credentials.

```
private void login()
{

   _service.CookieContainer = new CookieContainer();

   Passport passport = new Passport();
   RecordRef role = new RecordRef();

   passport.email = decrypt(myEncryptedEmail);
   passport.password = decrypt(myEncryptedPassword);
   passport.role = new RecordRef { internalId = "3" };
   passport.account = "1234567";

   SessionResponse resp = _service.login(passport);
}
```

After the requester has been successfully authenticated, a new session is created for that user. When using the login or ssoLogin operation to authenticate to NetSuite, user credentials are stored in HTTP headers and a JSESSIONID is assigned to every session.

If you are using inbound SSO, note also that the mapSSO operation has similar behavior. That is, although the primary purpose of the mapSSO operation is to create a mapping, it also logs the user in for a session. As part of this behavior, it returns a JSESSIONID.

If you are using application IDs, see Using Application ID with the Login and ssoLogin Operations.

Important: For session information to be successfully conveyed through a SOAP request, you must enable support for multiple cookie management in your application. For example, when using C#, you would include the following line:

```
service.CookieContainer = new CookieContainer();
```

Request-Level Credentials

Rather than authenticating to NetSuite by invoking the login operation, users have the option of sending their credentials in the SOAP header of each request. Sending credentials with each request eliminates the need for session management and separate logins. This approach may be beneficial to developers using PHP or other scripting languages that do not have built-in mechanisms for session management, manipulating HTTP headers, or tracking session IDs. Best practices to avoid concurrency errors include the use of request-level credentials, along with retries upon concurrency errors.

Note: You can use request-level credentials only when authenticating through user credentials or token-based authentication.

Note the following about using request-level credentials:

- To avoid errors, do not accept cookies when you use request-level credentials.
  - To prevent the acceptance of cookies in Java: use setMaintainSession(false).
  - To prevent the acceptance of cookies in C#, do not create a CookieContainer.
- Users must submit a role with every request.
- As with any request, only one web services request may be active at one time. Any attempt to send a second request results in a fault.

One way to use request-level credentials is to complete the following steps:

1. Assume that you have a class member called _service, which is an instance of the NetSuite service. For example:

```
namespace NetSuite
{

   class mainIntegration
      {
        public NetSuiteService _service;
                  ...
```

2. Create a passport method. The method should create a passport object that sets the user credential fields appropriately. It should also link this object to the _service object. For example:

```
 private void setPassport()
   {

      Passport passport = new Passport
      {
         email = decrypt(myEncryptedEmail);
         password = decrypt(myEncryptedEmail);
         account = "1234567",
         role = new RecordRef { internalId = "3" }
      };

      Console.WriteLine("Login info...");
      Console.WriteLine("   Email           : {0}", passport.email);
      Console.WriteLine("   Role Internal Id : {0}", passport.role.internalId);
      Console.WriteLine("   Account         : {0}\n", passport.account);

      _service.passport = passport;

}
```

   The previous example code lists the role internal ID of the user and the account ID. To find your account ID, go to the Web Services Preference page at Setup > Integration > Web Services Preferences. For a list of NetSuite role internal IDs, see Internal IDs Associated with Roles.

3. Use your passport method to modify the SOAP header that will be generated.

```
public SuiteTalkCourse()
{
      ServicePointManager.ServerCertificateValidationCallback += delegate { return true; };

      _service = new DataCenterAwareNetSuiteService();
      setPreferences();
      setApplicationInfo();
      setPassport();

}
```

4. The next time the application sends a request, the passport details are included in the header.

```
<soap:Header>
   <applicationInfo xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <applicationId>************************************</applicationId>
   </applicationInfo>
   <passport xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <email xmlns="urn:core_2017_1.platform.webservices.netsuite.com">my@email.com</email>
      <password xmlns="urn:core_2017_1.platform.webservices.netsuite.com">**********</password>

      <account xmlns="urn:core_2017_1.platform.webservices.netsuite.com">1234567</account>
      <role internalId="3" xmlns="urn:core_2017_1.platform.webservices.netsuite.com"/>
   </passport>
   <preferences xmlns="urn:messages_2017_1.platform.webservices.netsuite.com"/>
```

```
</soap:Header>
```

Important: If you use request-level credentials in a search request, you must use the searchMoreWithId operation to paginate through search results. See searchMoreWithId for more information.

Authorization for Web Services

Authorization is the process of ensuring that the requester has the appropriate entitlement to perform the requested operation. When users request to be authenticated, they also provide their NetSuite role. For every web services request, the system uses the role definition to ensure that the user has the required permission for the requested operation as well as the requested record type. The role must be provided in the Passport type via the login operation:

Example

```
<login xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<passport>
<ns1:email xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com">jblow@webservices.com
</ns1:email>
<ns2:password xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">mypassword<
   ns2:password>
<ns3:account xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">555555</ns3:account>

<ns4:role internalId="3" xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com"/>
</passport>
</login>
```

For detailed information on NetSuite roles and permissions and how the SuiteTalk implements roles and permissions rules, refer to Roles and Permissions in Web Services.

Session Management for Web Services

After you have been successfully authenticated using the login operation, a sessionID is created that must be passed to each subsequent request. Additional logins are not required as long as the session is active.

In NetSuite, session management includes:

- Session Timeouts
- Session Limits
- Manually Managing Cookies

Important: The following topics pertain to sessions that have been initiated using the login operation. If you authenticate to NetSuite by submitting your credentials in the SOAP header of your request should see Request-Level Credentials.

Session Timeouts

The Open Web Application Security Project (OWASP) provides the following guideline: All sessions should implement an absolute timeout, regardless of session activity. This timeout defines the maximum amount of time a session can be active. The session is closed and invalidated upon the defined absolute period. After the session is invalidated, the user must authenticate (log in) again in the web application and establish a new session. The absolute session timeout limits the amount of time possible for a potential attacker to use a hijacked session to impersonate a user.

Important: To enhance the security of your account, and to comply with the OWASP guideline, there is an absolute session timeout implemented for web services sessions. Currently, the absolute session timeout value is set to 60 minutes, even if the session is active.

If you use sessions with your web services integrations, you must ensure that your web services calls are able to handle session timeouts. It is recommended that your integrations should use sessionless protocols based on request-level credentials, such as user credentials or token-based authentication (TBA). Using TBA is recommended. See Authentication for Web Services for more information.

Web services sessions automatically time out after 20 minutes of inactivity, requiring a login to resume activity. If the server resubmits the cookie during the first 20 minutes, the inactivity timer is reset.

If a web services operation takes more than 15 minutes to complete, it automatically times out. In this case, consider using asynchronous request processing. For more information, see Synchronous Versus Asynchronous Request Processing.

Note: For security reasons, you should always send a logout operation as soon as your business operations are complete. Send a logout operation before the absolute session timeout or the inactivity session timeout.

If you explicitly log out of a session, and then attempt to use the same session, a SESSION_TIMED_OUT error message is returned. Your code should be prepared to handle session timeouts by retrying if an InvalidSessionFault (SESSION_TIMED_OUT) error is received.

Important: Do not make any integration decisions based on the value of the absolute timeout value or the idle session timeout values, as these values can be changed at any time without notice.

Session Limits

Session limits involve both user limits and account limits.

User Limits

A specific login (user name, account and password) that is used for a UI session can be used to create one or more sessions through web services by using the login operation. More web services sessions can occur concurrently, but the number of operations processed concurrently is limited. The UI session does not affect the web services sessions and operations.

Without a SuiteCloud Plus license, it is recommended for one user to only create one web services session using the login operation, and send requests using this session in one thread sequentially.

If the NetSuite server receives two login or other operations from the same user at the same moment, the second operation is rejected, and the error "Only one request may be made against a session at a time” is sent. The error indicates that the per user governance limit is exceeded.

Using the same login from multiple client applications, or multiple instances of the same application is NOT supported.

If you have a SuiteCloud Plus license, you can designate users as concurrent web services users. One SuiteCloud Plus license allows you to designate one user as concurrent web services user. One concurrent web services user can send up to 10 concurrent web services requests.

Note: If the total account limit for concurrent requests is exceeded, one concurrent web services user might not be able to send 10 concurrent requests.

Account Limits

Note that starting from version 2017.2, web services concurrency is not only governed separately per user and authentication method, but also per account. For information about account concurrency limits, see Web Services and RESTlet Concurrency Governance Starting From Version 2017.2.

Each SuiteCloud Plus license increases the account concurrency governance limit by 10, meaning that you can have 10 additional parallel requests at the same per account. The account base limit depends on your service tier. For details, see Enabling Web Services Concurrent Users with SuiteCloud Plus.

Manually Managing Cookies

When submitting web services requests, the NetSuite server can accept any SOAP document as long as it is valid against the NetSuite WSDL, and it does not matter how the SOAP document was generated. You must ensure that all cookies received from the server are managed and passed back to the server in subsequent calls. There are a total of three cookies that are initially returned in the HTTP header of the login response, including the JSESSIONID.

Note: The JSESSIONID cookie is only generated if you use the Login or the ssoLogin operation for authentication. If you use request-level credentials, this cookie is not generated.

Ensure that your client persists any cookies that our server sends you (re-submits the cookie on the next request). In Axis, this is accomplished by enabling **MaintainSession** on the generated Binding object. In .NET, this involves allocating a CookieContainer. If persistence is not maintained, you will receive the following error message: SESSION_TIMED_OUT — ”Your connection has timed out. Please log in again.”

Example

The following code shows the cookie information returned in the login response. The cookie information is contained in the Set-Cookie lines of the code.

```
HTTP/1.1 200 OK
Date: Mon, 16 Oct 2017 13:34:48 GMT
Content-Type: text/xml; charset=utf-8
X-N-OperationId: a8f84e6e-6300-4f92-9a1a-30fae3da1978
X-Netsuite-Approute: host=test.eng.netsuite.com;enter=2017-10-16 06:34:48.716;buf=785;versrc=pa
ssport;apco=err_cxn;ver=2018.1.0;dcid=000;exit=2017-10-16 06:34:48.720;events=setcookie,rule
Set-Cookie: JSESSIONID=iR8lZOFn88GFV23ZVnnS0BUTi6V2fJZZzil02Xf9gyXvNTVNarxfIWRanvswN0LDYW-gKsBU
o75gVslo42EfwTm-qlKnTDo5SynW_nc-Yvh0dQJVTzHokW9qGvoyn0jW!-477725890; path=/; HttpOnly
Set-Cookie: NS_ROUTING_VERSION=LAGGING; path=/
Set-Cookie: NS_VER=2017.2.0; domain=test.eng.netsuite.com; path=/
P3P: CP="CAO PSAa OUR BUS PUR"
Vary: User-Agent
Connection: close
```

For each subsequent request, all of the cookies must be returned in the HTTP header to maintain the session as follows:

C#

```
NetSuiteService nss = new NetSuiteService();
nss.Url = https://webservices.netsuite.com/services/NetSuitePort_2017_1_0 ;
nss.CookieContainer = new CookieContainer();
```

Note: This snippet assumes that your integration includes a `Using System.Net;` statement.

Java with Axis 1.4

```
NetSuiteServiceLocator service = new DataCenterAwareNetSuiteServiceLocator(_properties.getPrope
rty("login.acct"));
service.setMaintainSession(true);
```

Encryption for Web Services

NetSuite's web services are protected by TLS encryption. NetSuite supports TLS 1.0, 1.1, and 1.2. Only requests sent using TLS encryption are granted access.

Custom Field Security

Custom field security can be applied on a per-field basis. If field security has been applied to a field in the UI, the custom field schema will include the field level security metadata. Consequently, users should be aware that the permissions they specify on custom fields will apply to any existing integration that is in place using endpoints 2008_1 and older. Certain field permissions such as NONE and VIEW might break the integration in an unintended way.

Note: See the help topic Restricting Access to Custom Fields for details on this feature.

If the custom field security feature is turned on, web services will respect the access level set on each field. For example, if you have set a field permission to NONE, and you have code that references the field, you will not be able to read or write to that field in your code. NetSuite web services will essentially ignore that field when the SOAP request is sent. Similarly, if a field permission has been set to VIEW, you will be able to read the field's value, but you will be unable to set the field's value.

Web services developers should keep custom field security in mind when designing their integrations. Before beginning a project, they should work with their company's NetSuite administrator to review which fields may have custom security applied.

Platform Features

The SuiteTalk platform refers to the software infrastructure used to expose NetSuite application functionality via web services. This section describes some of the core features of the SuiteTalk and includes the following sections:

- Enabling Web Services Concurrent Users with SuiteCloud Plus: provides an overview of the Web Services Plus license type.
- Operations and Their Internal ID Requirements: describes IDs and how they are used.
- Invoking a UI Session for an External Application: describes how to submit an HTTPS POST for single login between external web services applications and the NetSuite UI.
- Using Web Services to Send Data from a CSV File describes how to use the NetSuite Data Loader tool.
- Searching for Web Services Log Information: describes how to search for web services operation and record information.

Important: NetSuite hosts customer accounts in multiple data centers. For that reason, the correct URL for web services access varies depending on the data center hosting the account. Your integration **must** incorporate logic that dynamically determines the correct URL. With the 2012.2 and later endpoints, you should use the getDataCenterUrls operation to dynamically discover the correct URL. With older endpoints, you should use the REST roles service. For details, see the help topic Using the REST roles Service to Get User Accounts, Roles, and Domains. Starting from version 2017.2, you can also use the DataCenterUrls REST Service.

Enabling Web Services Concurrent Users with SuiteCloud Plus

Your account must have the web services feature enabled before you can assign a user to be a web services concurrent user. See Enabling the Web Services Feature for details.

Any user can be designated as concurrent web services user if one or more SuiteCloud Plus license has been purchased.

Generally, you can purchase one SuiteCloud Plus license per NetSuite account. Contact your sales representative to discuss any need for additional licenses.

Prior to version 2017.2, if you had one or more SuiteCloud Plus license, each designated concurrent web services user was guaranteed 10 concurrent web services requests.

Starting from version 2017.2, web services and RESTlet concurrency is also governed per account. The new account governance limit applies to the combined total of web services and RESTlet requests. Due to this change, if a designated user makes 10 concurrent web services requests, the number of successful requests depends on other web services or REST requests that are being processed. Each SuiteCloud Plus license increases the account base limit for concurrent web services requests by 10. The account base limit depends on your service tier.

Note: The account limit is lenient, and it is possible to temporarily exceed the limit.

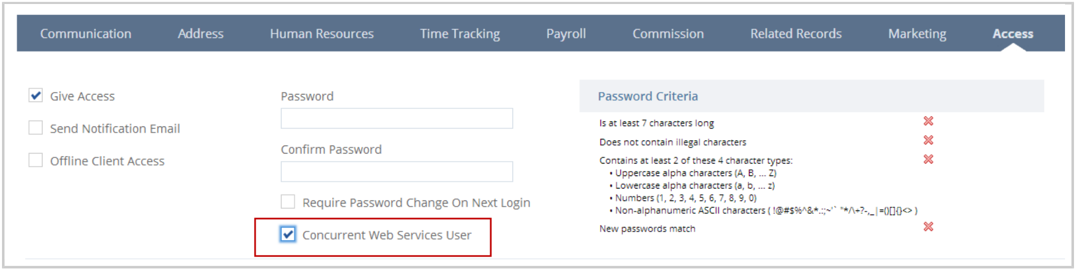
Important: If you use token-based authentication with web services, a single user can send concurrent requests up to the account limit. For more information, see Web Services Governance for Token-Based Authentication.

For more information about account base limits and the concurrency governance, see Web Services and RESTlet Concurrency Governance Starting From Version 2017.2.

Note: SuiteCloud Plus provides other benefits. It increases the number of scheduled script processors. For details, see the help topic Scheduled Scripts on Accounts with Multiple Processors (SuiteCloud Plus). It allows the use of multi-threading for CSV import jobs and increases the number of import jobs that can be run in parallel. For details, see the help topic Use Multiple Threads and Multiple Queues to Run CSV Import Jobs. For a summary of SuiteCloud Plus capabilities, see the help topic SuiteCloud Plus Settings.

To designate a user as a concurrent web services user:

1. Open the employee record.
2. Click the **Access** subtab.
3. Check the **Concurrent Web Services User** box.
4. Click **Save**.



☑ Give Access
☐ Send Notification Email
☐ Offline Client Access

Password

Confirm Password

☐ Require Password Change On Next Login
☑ Concurrent Web Services User

Password Criteria

Is at least 7 characters long ✗
Does not contain illegal characters ✗
Contains at least 2 of these 4 character types: ✗
- Uppercase alpha characters (A, B, ... Z)
- Lowercase alpha characters (a, b, ... z)
- Numbers (1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
- Non-alphanumeric ASCII characters (!@#$%^&*.:;~'` "*/\+?-,_|=()[]{}<>)

New passwords match ✗

Operations and Their Internal ID Requirements

The following table lists the ID requirements for each operation. Note that IDs are only required in calls where specific records corresponding to the call exist in the NetSuite database. For example, on an add operation, no ID is required in the request since the record does not yet exist.

An internal ID is created by the system and returned in the response. On search operations, because a specific record is not being called, an internal ID is not required in the request. However, an internal ID for each record found is returned in the response.

Note: Since external IDs are provided by the client application, if an external ID is desired for a record, it can be submitted on an Add operation.

Operation	Record ID Required in Request	Record ID Returned in Response
login	n/a	n/a

Operation	Record ID Required in Request	Record ID Returned in Response
logout	n/a	n/a
add/addList	No	Yes
update/updateList	Yes	Yes
delete/deleteList	Yes	Yes
getDeleted	No	Yes
get/getList	Yes	Yes
getAll	No	Yes
search/searchNext/searchMore/ searchMoreWithId	No	Yes
getSavedSearch	No	Yes

The following samples show the use of internal IDs as they pertain to operations. Click to see samples for any of the following:

- add operation
- update operation
- deleteList operation
- getList operation
- search operation

add operation

For an add operation, an internal ID is *not* required in the request since the record does not yet exist. The internal ID (in this case 100101) is returned in the response.

Note: There are internal IDs listed in the request, but these internal IDs are *embedded* in the request and do not correspond to the actual record being added (an event) but to other existing records associated with the event record being added.

Request

```
<soap:Body>
<platformMsgs:add>
   <platformMsgs:record xsi:type="actSched:CalendarEvent">
      <actSched:title>Web Services Meeting</actSched:title>
      <actSched:organizer internalId="-5" type="calendarEvent" xsi:type="platformCore:RecordRef
">
         <platformCore:name>Mr. Wolfe</platformCore:name>
      </actSched:organizer>
      <actSched:location>Main Conference Room</actSched:location>
      <actSched:attendeeList replaceAll="true" xsi:type="actSched:CalendarEventAttendeeList">
         <actSched:attendee xsi:type="actSched:CalendarEventAttendee">
            <actSched:sendEmail>false</actSched:sendEmail>
            <actSched:attendee internalId="21" type="calendarEvent"
            xsi:type="platformCore:RecordRef"/>
            <actSched:response>_accepted</actSched:response>
            <actSched:attendance>_optional</actSched:attendance>
         </actSched:attendee>
         <actSched:attendee xsi:type="actSched:CalendarEventAttendee">
            <actSched:sendEmail>false</actSched:sendEmail>
            <actSched:attendee internalId="27" type="calendarEvent"
            xsi:type="platformCore:RecordRef"/>
            <actSched:response>_accepted</actSched:response>
            <actSched:attendance>_optional</actSched:attendance>
         </actSched:attendee>
      </actSched:attendeeList>
   </platformMsgs:record>
</platformMsgs:add>
</soap:Body>
```

Response

```
<soapenv:Body>
   <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <writeResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
         <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.
com"/>
         <baseRef internalId="100101" type="calendarEvent" xsi:type="ns2:RecordRef"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

update operation

In this example the record added above is being updated — since the internal ID matches the one created in the add operation (100101). Here, the internal ID and the record type are required in the request and both are also returned in the response.

Request

```
<soap:Body>
   <platformMsgs:update>
      <platformMsgs:record internalId="100101" xsi:type="actSched:CalendarEvent">
         <actSched:title>Web Services Meeting (Platform)</actSched:title>
      </platformMsgs:record>
   </platformMsgs:update>
</soap:Body>
```

Response

```
<soapenv:Body>
   <updateResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <writeResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
```

```
         <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.
com"/>
         <baseRef internalId="100101" type="calendarEvent" xsi:type="ns2:RecordRef"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </updateResponse>
</soapenv:Body>
```

deleteList operation

In the following delete request, the internal IDs are required for the request and returned in the response. In this case, three event records are deleted (100101, 100102, and 100103).

Request

```
<soap:Body>
<platformMsgs:deleteList>
   <platformMsgs:baseRef internalId="100101" type="calendarEvent" xsi:type="platformCore:Record
Ref"/>
   <platformMsgs:baseRef internalId="100102" type="calendarEvent" xsi:type="platformCore:Record
Ref"/>
   <platformMsgs:baseRef internalId="100103" type="calendarEvent" xsi:type="platformCore:Record
Ref"/>
</platformMsgs:deleteList>
</soap:Body>
```

Response

```
<soapenv:Body>
<deleteListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<writeResponseList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <writeResponse>
      <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com
"/>
      <baseRef internalId="100101" type="calendarEvent" xsi:type="ns2:RecordRef"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
   </writeResponse>
   <writeResponse>
      <ns3:status isSuccess="true" xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com
"/>
      <baseRef internalId="100102" type="calendarEvent" xsi:type="ns4:RecordRef"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com"/>
   </writeResponse>
   <writeResponse>
      <ns5:status isSuccess="true" xmlns:ns5="urn:core_2017_1.platform.webservices.netsuite.com
"/>
      <baseRef internalId="100103" type="calendarEvent" xsi:type="ns6:RecordRef"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.com"/>
   </writeResponse>
</writeResponseList>
</deleteListResponse>
</soapenv:Body>
```

getList operation

In this example, an internal ID is called for each record to be retrieved — in this case, three different event records. Again, the internal ID is required for the request and returned in the response.

Request

```
<soap:Body>
<platformMsgs:getList>
   <platformMsgs:baseRef internalId="100104" type="calendarEvent" xsi:type="platformCore:Record
Ref"/>
   <platformMsgs:baseRef internalId="100105" type="calendarEvent" xsi:type="platformCore:Record
Ref"/>
   <platformMsgs:baseRef internalId="100106" type="calendarEvent" xsi:type="platformCore:Record
Ref"/>
</platformMsgs:getList>
</soap:Body>
```

Response

```
<soapenv:Body>
<getListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<readResponseList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<readResponse>
   <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"/>

   <record internalId="100104" xsi:type="ns2:CalendarEvent"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ns2="urn:scheduling_2017_1.activities.webservices.netsuite.com">
      <ns2:title>Customization Meeting</ns2:title>
      <ns2:organizer internalId="-5">
      .........[more fields]
   </record>
</readResponse>
<readResponse>
   <ns8:status isSuccess="true" xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com"/>

   <record internalId="100105" xsi:type="ns9:CalendarEvent"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ns9="urn:scheduling_2017_1.activities.webservices.netsuite.com">
      <ns9:title>Web Services Meeting (Records)</ns9:title>
      <ns9:organizer internalId="-5">
      .........[more fields]
   </record>
</readResponse>
<readResponse>
   <ns15:status isSuccess="true" xmlns:ns15="urn:core_2017_1.platform.webservices.netsuite.com"
```

```
/>
   <record internalId="100106" xsi:type="ns16:CalendarEvent"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ns16="urn:scheduling_2017_1.activities.webservices.netsuite.com">
      <ns16:title>Web Services Meeting</ns16:title>
      <ns16:organizer internalId="-5">
      .........[more fields]
   </record>
</readResponse>
</readResponseList>
</getListResponse>
</soapenv:Body>
```

search operation

For the search operation, an internal ID is not required for the request but is returned for each record found that matches the specified criteria. In this case, two event records are returned for a search request calling for each event record that contains **web services** in the title field.

(Request)

```
<soap:Body>
<platformMsgs:search>
   <platformMsgs:searchRecord xsi:type="actSched:CalendarEventSearch">
      <actSched:title operator="contains" xsi:type="platformCore:SearchStringField">
         <platformCore:searchValue>web services</platformCore:searchValue>
      </actSched:title>
   </platformMsgs:searchRecord>
</platformMsgs:search>
</soap:Body>
```

(Response)

```
<soapenv:Body>
<searchResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<searchResult xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
   <status isSuccess="true"/>
   <totalRecords>2</totalRecords>
   <pageSize>10</pageSize>
   <totalPages>1</totalPages>
   <pageIndex>1</pageIndex>
   <recordList>
      <record internalId="100105" xsi:type="ns1:CalendarEvent"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:ns1="urn:scheduling_2017_1.activities.webservices.netsuite.com">
         <ns1:title>Web Services Meeting (Records)</ns1:title>
         <ns1:organizer internalId="-5">
         .....[more fields]
      </record>
      <record internalId="100106" xsi:type="ns2:CalendarEvent"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:ns2="urn:scheduling_2017_1.activities.webservices.netsuite.com">
         <ns2:title>Web Services Meeting</ns2:title>
         <ns2:organizer internalId="-5">
         .....[more fields]
      </record>
   </recordList>
</searchResult>
</searchResponse>
</soapenv:Body>
```

Invoking a UI Session for an External Application

As an alternative to using SuiteTalk web services, you can allow an external application to access NetSuite by submitting a POST to a specific wslogin page URL. To use this approach, you submit the appropriate credentials and a taskId identifying the desired page. The wslogin page then authenticates the request, provides a new session, and redirects to the requested page.

This feature is **NOT** intended to support any type of user access to NetSuite. It is intended for use solely as part of an integration strategy. For example, you could use this feature in conjunction with an external web services application that moves data into NetSuite. The external application would need to have user credentials but, again, this method is not recommended as a means of providing NetSuite access to actual users.

The following table lists the POST parameters that can be submitted.

Parameter	**Type**	**Required for Authentication**	**Description**
email	text	Yes	Must reflect a valid email address of an entity in your account.
password	password	Yes	The password of the entity associated with the email provided.
taskid	text	Yes	To find the TaskID for a specific page when in NetSuite, view the HTML page source of the page and search for the `nlPopupHelp` string. For example, this search might return `onclick="nlPopupHelp('LIST_SCRIPT','help')`, where `LIST_SCRIPT` is the task ID.
id	text	Yes	The ID of a record type available in your NetSuite account.
role	text	Yes, unless the entity (user) has a default role specified on the View All Roles page	Sets the role for the login. This must be a valid role for the entity (user) logging in.
e	text	Optional	If set to T, the record is displayed in edit mode. Any other value is ignored and causes the record to be displayed in view mode.

The post URL is —

```
https://system.netsuite.com/app/webservices/wslogin.nl?c=######
```

— where ###### is your account number.

Because the URL associated with the POST mechanism begins with https, the credentials supplied will be encrypted and, therefore, will NOT be vulnerable during transmission.

Note: Initiating a new browser session using this mechanism will invalidate an existing session that might have been invoked elsewhere using the same credentials.

Using Web Services to Send Data from a CSV File

If you have external data stored in a CSV file and want to load this data into NetSuite, you can use the new NetSuite Data Loader sample application. With this application, you can send web services requests that use the data in your CSV files. This approach is beneficial because it lets you use parallel connections, as available based on your NetSuite licensing.

Important: NetSuite Data Loader is intended as a demo application. No regular maintenance is provided for this application.

NetSuite Data Loader

NetSuite Data Loader is a command line Java application. It requires Gradle and Java JDK 6 to build. The source code is available for download.

To use the NetSuite Data Loader, go to the SuiteTalk tools download page, click the link for the .zip file, and agree to the license terms. After the tool has been downloaded, navigate to the tool's doc directory, and review the README file for more detailed instructions.

Important: To access the download page, you must be logged in to NetSuite as administrator, and the Web Services feature must be enabled in your account.

Searching for Web Services Log Information

Beginning in 2016.2, you can access web services log information through searches in the NetSuite UI. You can set up results columns and filter data to return only the details you need. These searches support analysis of web services failures.

You can create web services log searches by going to Reports > New Search and selecting one of the following new search types:

- **Web Services Operations** — Use this type to search for and analyze information about failing web services requests and operations.
- **Web Services Record Processing** — Use this search type to search for and analyze information about failing web services records.

Each new search type includes join fields to the other search type, so the two search types can be joined for a more comprehensive web services log information search.

Note the following:

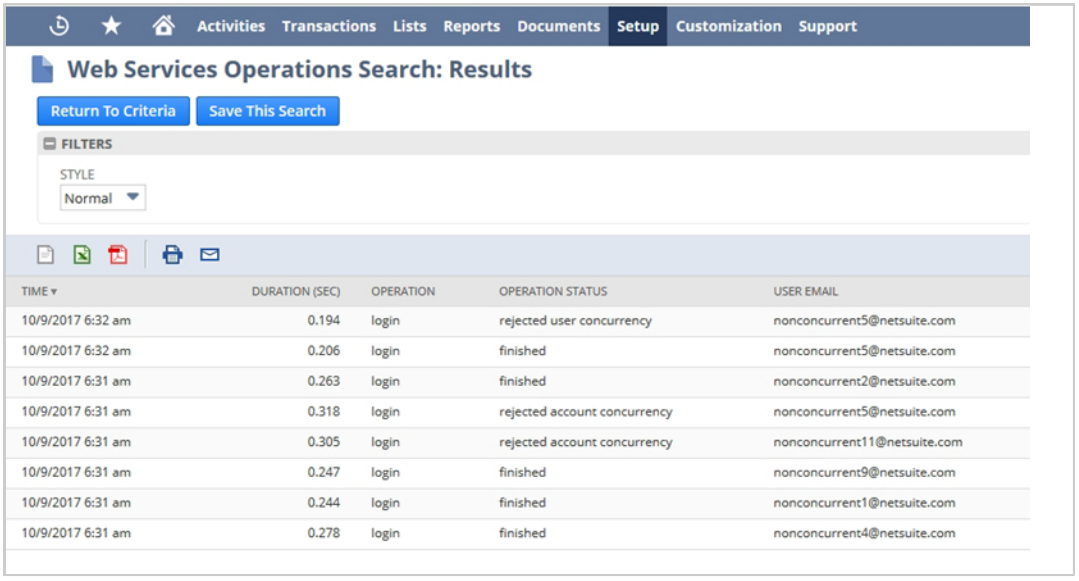
- Web services execution log items are retained for one month, so searches do not return data older than one month.
- Results for these searches are limited to log items created as of 2016.2; log items created prior to this release are not included.

Web Services Operations Search Type

The following web services operation fields are available for search:

- **API Version** — The API version used to perform the operation.
- **Authentication Method** — The authentication method used by the operation.
- **Job ID** — The Job ID.
- **Time** — The date and time the operation was performed.
- **Duration** — The length of time (in seconds) it took to process the operation.
- **Operation** — The web services operation.
- **Operation Status** — The operation status; failed, finished, finished with errors, processing, or rejected.
- **User Email** — The email address associated with the account that performed the operation.
- **Records In Request** — The number of records in the request.
- **Records in Response** — The number of records in the response.
- **Failed Records** — The number of records that failed.
- **Service Type** — The type of service used: SYNC or ASYNC.
- **Integration** — The related integration record.
- **Finished Records** — The number of records completed for the selected operation. This column is available for advanced searches.
- **Request** — A link to the SOAP request. This column is available for advanced searches.
- **Response** — A link to the SOAP response. This column is available for advanced searches.

The following image shows the results of a web services operations search.



Web Services Operations Search: Results

STYLE: Normal

TIME ▾	DURATION (SEC)	OPERATION	OPERATION STATUS	USER EMAIL
10/9/2017 6:32 am	0.194	login	rejected user concurrency	nonconcurrent5@netsuite.com
10/9/2017 6:32 am	0.206	login	finished	nonconcurrent5@netsuite.com
10/9/2017 6:31 am	0.263	login	finished	nonconcurrent2@netsuite.com
10/9/2017 6:31 am	0.318	login	rejected account concurrency	nonconcurrent5@netsuite.com
10/9/2017 6:31 am	0.305	login	rejected account concurrency	nonconcurrent11@netsuite.com
10/9/2017 6:31 am	0.247	login	finished	nonconcurrent9@netsuite.com
10/9/2017 6:31 am	0.244	login	finished	nonconcurrent1@netsuite.com
10/9/2017 6:31 am	0.278	login	finished	nonconcurrent4@netsuite.com

Note: In production environments, the data is accessible for 21 days. In sandbox environments, the data is accessible for seven days.

Web Services Record Processing Search Type

The following web services record fields are available for search:

- **Duration** — The length of time (in seconds) it took to process the record.
- **Operation** — The operation performed on the record, either ADD, UPDATE, or DELETE.
- **Record Type** — The record type.
- **Sequence Number in Request** — When multiple records are processed during an operation, this field indicates the sequence number or processed record within the operation.
- **Status** — The record’s status, either Complete, Partially complete, or Failed.
- **Time** — The date and time when the record was processed.
- **Record** — The record that was processed. This column is available for advanced searches.

Note: In production environments, the data is accessible for 21 days. In sandbox environments, the data is accessible for seven days.

Types

This section describes the following:

- Built-in Types — primitive or derived datatypes defined in the XML Schema specification
- Complex Types — NetSuite derived datatypes
- Custom Field Types — datatypes used for custom fields
- Search Types — datatypes used for search records
- Platform Enumerations — datatypes used to populate system defined lists

A WSDL document in web services uses the XML Schema to define the structure, semantics and constraints of the messages sent between the sender and recipient. The XML Schema is a W3C standard.

There is a strong analogy between object-oriented programming and XML Schema. A type defined in XML Schema can be used to represent an instance of that type in an XML document. Therefore, the body of a SOAP message consists of a combination of such instances of an XML Schema type.

Built-in Types

Built-in (or primitive) types are those that are not defined in terms of other datatypes. They are used as a standardized way to define, send, receive and interpret basic data types in SOAP messages. Primitive data types used in the SuiteTalk can be modified for display purposes. For example, although a price field may be passed in the SOAP messages using an integer primitive data type, the NetSuite UI may format the value with a currency symbol for display purposes.

Of the extensive set of built-in (or primitive) types provided by the XML Schema language specification, the SuiteTalk implementation uses the following built-in types. For detailed information on XML Schema built-in types, refer to http://www.w3.org/TR/xmlschema-2/.

- **string** —represents character strings in XML
- **int** — derived from the decimal type and represents the standard concept of the integer numbers
- **double** — is patterned after the IEEE double-precision 64-bit floating point type
- **boolean** — represents the concept of binary-valued logic: {true, false}. In NetSuite, a boolean field can be set to true, false, 1 to indicate true, or 0 to indicate false.
  - In the 2010.2 endpoint and beyond, SuiteTalk validates boolean values. Each value must be a true xsd boolean (0,1,true,false) as per the W3C specification. For details, see the specification.
  - In endpoints prior to 2010.2, invalid values are treated as false.
- **dateTime** — represents integer-valued year, month, day, hour and minute properties. For example, **2005-09-21T15:24:00.000-07:00**, where 2005-09-21 is the date, 15:24:00.000 is the time and -07:00 is the timezone offset.

Using The dateTime Type in NetSuite

When setting a date field using web services, a datetime value must be entered. The time and time zone following the date are used to determine the resulting date. When getting the date field through web services, the user's preferred time zone influences the resulting datetime, i.e. midnight in the user's time zone of the stored date (which is also displayed in the UI) is converted to the PST/PDT and sent to the web services user.

> **Note:** During winter months, the Pacific Timezone is 8 hours behind Greenwich Mean Time (GMT-8) and is referred to as PST. In summer months it is 7 hours behind Greenwich Mean Time (GMT-7) and is referred to as PDT.

The following table gives several examples:

Datetime set in date field	Time Zone of the User	Date shown in UI	Datetime when getting the record [2]
2013-02-21T00:00:00.000-06:00	GMT-06:00	2/21/2013	2013-02-20T22:00:00.000-08:00
2013-08-21T00:00:00.000-05:00	GMT-05:00 [1]	8/21/2013	2013-08-20T22:00:00.000-07:00
2013-02-21T00:00:00.000-05:0	GMT-08:00	2/20/2013 [3]	2013-02-20T00:00:00.000-08:00
2013-08-21T00:00:00.000-07:00	GMT-05:00 [1]	8/21/2013	2013-08-20T22:00:00.000-07:00

Notes for the example table:

1. The user has set their time zone to GMT-6, but the DST affects the result.
2. This is midnight of the date from the UI that is converted to the PST/PDT. Please note the time difference, which equals the difference between the user's preferred time zone and the PST/PDT.
3. The datetime which is to be saved is not greater than midnight in the user's preferred time zone.

Guidelines for The dateTime Type

Consider the following for the guidelines for dateTime type:

- To avoid problems with conversions, set the time zone of the web services employee to "(GMT-08:00) Pacific Time (US & Canada)" in Home > Set Preferences. Also, submit the datetimes for date fields as midnight and in PST/PDT (see the Setting Proper Date/Time section below). Doing this will result in no conversions or truncations (what is submitted will be identical to what is retrieved with web services).
- When changing time zone preferences the user must re-log using web services for the change to take effect. If the same session is used the old time zone will be used as well. Caching may also occur for a period after changing time zones.
- Users setting the field value need to pay attention to the DST offset, as it influences the decision point (midnight of user's time zone) according to which value is set to the field.
- If the time zone specified in set field request is different than the time zone of the user setting the field, the datetime is first converted to the time zone of the user and the date portion is stored. The time is then set to midnight and returned in PST/PDT. See the second row of the preceding example table for an example.
- When posting dateTime values to NetSuite, it is recommended that you indicate timezone. If no timezone is indicated, the “PST/PDT” timezone is used (note that for backward compatibility reasons NetSuite does not use GMT).

Setting Proper Date/Time Format for Web Services

Setting of Date Time, Date or Time fields for Web Services using Date-Time-Timezone: <xmp><s0:endDate>2013-10-30T20:01:24.405-08:00</s0:endDate></xmp>

If the user does not indicate the Timezone on the request, Netsuite uses its own system Timezone (PST UTC-8/PDT UTC-07): <xmp><s0:endDate>2013-10-30T20:01:24.405</s0:endDate></xmp>

Users can indicate an UTC Timezone by adding the 'Z' character at the end of the DateTime Format: <xmp><s0:endDate>2013-10-30T20:01:24.405Z</s0:endDate></xmp>

Complex Types

Complex types are structured types built by aggregating elements of simple or previously defined complex types. NetSuite complex types are defined in the platform core and platform core types XSDs. All NetSuite record types ultimately reference one of these complex types.

NetSuite complex types include the following:

- Passport
- TokenPassport
- Record
- RecordList
- BaseRef
- RecordRef
- CustomRecordRef
- ListOrRecordRef
- Status
- StatusDetail
- NullField
- ReadResponse
- ListReadResponse
- ListWriteResponse
- WriteResponse

Passport

The Passport type is used by the login operation to encapsulate the security credentials required for authentication. The passport type is defined in the core.xsd.

Field Name	XML Schema Type	Req	Notes
email	xsd:string	Y	
password	xsd:string	Y	
account	xsd:string	Y	Must correspond to a valid NetSuite account number
role	RecordRef	Y	A role is a set of permissions that lets a user access specific areas of data. If not set, NetSuite uses the default Web Services role set for this user.

The default web services role for a specific user is dependent on the values set when setting the permissions for a particular role. These are set through the UI in Setup > Users/Roles > Manage Roles.

TokenPassport

The TokenPassport type is used by web services requests that use token-based authentication. The TokenPassport type is defined in the core.xsd.

Field Name	XML Schema Type	Req	Notes
account	xsd:string	Y	
consumerKey	xsd:string	Y	
token	xsd:string	Y	
nonce	xsd:string	Y	
timestamp	xsd:long	Y	
signature	xsd:string	Y	

Note: For information about the TokenPassport type, see TokenPassport Complex Type.

Record

The Record type is an abstract type used as the parameter for the add, addList, delete, deleteList, update and updateList operations. It is also returned in the get, getList, search, searchMore and searchNext operations. All business object types extend the Record type. The record type is defined in core.xsd.

Field Name	XML Schema Type	Req	Notes
id	xsd:string (attribute)	N	
xsi:type	xsi:type (attribute)	N	This is a field (attribute) that is automatically implemented by the XML Schema. The value should represent the concrete Record type such as Customer or Event.
nullFieldList	NullFields[]	N	A list of fields that are to be set to null explicitly in the update operation. **Note:** You cannot set the Custom Form field to NULL. Any request to set this field to NULL is ignored.

RecordList

The RecordList type is an array of the Record type. The recordList type is defined in core.xsd.

Field Name	XML Schema Type	Req	Notes
record	Record []	N	

BaseRef

The BaseRef type is an abstract type used to reference any existing record in NetSuite including other business records and custom records. The BaseRef type is defined in core.xsd.

Field Name	XML Schema Type	Req	Notes
name	BaseRef	N	

RecordRef

The RecordRef type is used to reference an existing record in NetSuite in get operations. Typically, a RecordRef references one of the following:

- Another business object such as a customer or sales order
- An entry in a user defined list such as the category list for contacts or sales tax items

The recordRef type descends from BaseRef and is defined in core.xsd.

Field Name	XML Schema Type	Req	Notes
internalId	xsd:string (attribute)	Y	See Using Internal IDs, External IDs, and References.
type	xsd:string (attribute)	N	Reference to a value in a web services only system list. If the type is known by context, the type is NOT required.
name	xsd:string	N	This is a read-only field that is populated by NetSuite when it's a part of a get or search response. If this field is populated during a write operation, it will be ignored.

CustomRecordRef

The CustomRecordRef type is used to reference any existing custom record in NetSuite. The CustomRecordRef type descends from BaseRef and is defined in core.xsd.

Important: Setting the RecordRef.type to **customRecord** on an add does NOT return a CustomRecord. You must use CustomRecordRef. The CustomRecordRef has a typeId to indicate which kind of CustomRecord it is.

Field Name	XML Schema Type	Req	Notes
internalId	xsd:string (attribute)	Y	References the primary record internal Id. This Id corresponds to the type of custom record. For a list of possible values, see ListOrRecordRef.
typeId	xsd:string	Y	References the custom record type Id.
type	xsd:string (attribute)	N	Reference to a value in a web services only system list.
name	xsd:string	N	

ListOrRecordRef

The listOrRecordRef type is defined in core.xsd.

Field Name	XML Schema Type	Req	Notes
internalId	xsd:string (attribute)	Y	See Using Internal IDs, External IDs, and References.
externalId	xsd:string (attribute)	N	Use to reference records by their external ID in select and multi-select custom fields.
typeId	xsd:string (attribute)	N	Reference to a value in a web services only system list.
name	xsd:string	N	This is a read-only field that is populated by NetSuite when it's a part of a get or search response. If this field is populated during a write operation, it is ignored.

Each record type in NetSuite has a corresponding internal ID (or typeId). This internal ID is required when using ListOrRecordRef since the type of record being referenced needs to be specified.

Note: To update a ListOrRecordRef object, you must use either the internal ID or the external ID in your request. Using the name field only does not update the object.

For example, in the following code a new ListOrRecordRef object is created. The list references a specific Entity record as designated by the internalId of **1011** and specifies that the record is of the type **customer (-2)**. Note that customer records have an internal ID of -2 as shown in the table below.

```
ListOrRecordRef[] fieldNameEntity = new ListOrRecordRef[1];
fieldNameEntity[0] = new ListOrRecordRef();
fieldNameEntity[0].setInternalId("1011");
fieldNameEntity[0].setTypeId("-2");
```

Status

The Status type contains an array of objects of type StatusDetail. The status type is defined in core.xsd.

Field Name	XML Schema Type	Req	Notes
statusDetail	StatusDetail []	N	Used to capture the specific details for the status. See StatusDetail.
isSuccesss	xsd:Boolean (attribute)	Y	Indicates whether the status is successful or not. If false, one or more statusDetail objects are populated.

StatusDetail

The StatusDetail type is used to capture the specific details for the status. The statusDetail type is defined in core.xsd.

Field Name	XML Schema Type	Req	Notes
code	xsd:string	Y	The status code. See Web Services Error Handling and Error Codes for a listing of codes.
message	xsd:string	Y	The detailed message for this status. See Web Services Error Handling and Error Codes for details.
type	xsd:string		Reference to a value in a web services only system list. Values: error, warning See Web Services Error Handling and Error Codes for details.

NullField

The NullField type is defined in core.xsd. It contains the following fields.

Field Name	XML Schema Type	Req	Notes
name	xsd:string	Y	Name of the field to be null. The specified name must exactly match an existing field name.

Note: You cannot set the Custom Form field to NULL. Any request to set this field to NULL is ignored.

ReadResponse

The ReadResponse type is used by the following read operations.

- The getResponse output message for the get operation.
- The searchResponse output message for the search operations.

These types have a field named readResponse of the type ReadResponse. The ReadResponse type is defined in message.xsd.

Field Name	XML Schema Type	Req	Notes
status	Status	Y	
recordRef	RecordRef	N	

ListReadResponse

The ListReadResponse type is used by the following operations.

- The GetListResponse output message for the getList operation.

These types have a field named response of type ListReadResponse. The ListReadResponse type is defined in message.xsd.

Field Name	XML Schema Type	Req	Notes
response	ReadResponse[]	Y	An array of ReadResponse types.

ListWriteResponse

The ListWriteResponse type is used by the following operations.

- The AddListResponse output message for the addList operation.
- The UpdateListResponse output message for the updateList operation.
- The DeleteListResponse output message for the deleteList operation.

These types have a field named response of type ListWriteResponse. The ListWriteResponse type is defined in message.xsd.

Field Name	XML Schema Type	Req	Notes
response	WriteResponse[]	Y	An array of WriteResponse types.

WriteResponse

The WriteResponse type is used by the following operations:

- The AddResponse output message for the add operation
- The UpdateResponse output message for the update operation
- The DeleteResponse output message for the delete operation

These types have a field named writeResponse of type WriteResponse. The WriteResponse type is defined in message.xsd.

Field Name	XML Schema Type	Req	Notes
status	Status	Y	
recordRef	RecordRef	N	

Custom Field Types

Custom fields are represented by the type CustomFieldRef, which is an abstract type. The table below contains a list of concrete custom field types that extend the CustomFieldRef type. Each type is followed by its corresponding type in the UI.

XML Schema Type	Custom Field Type in UI
LongCustomFieldRef	Integer
DoubleCustomFieldRef	Decimal Number
BooleanCustomFieldRef	Check Box
StringCustomFieldRef	Free-Form Text Text Area Phone Number E-mail Address Hyperlink Rich Text
DateCustomFieldRef	Date Time of Day

XML Schema Type	Custom Field Type in UI
	or Date/Time (both in one field)
SelectCustomFieldRef	List/Record Document
MultiSelectCustomFieldRef	Multiple Select

CustomFieldRef

The CustomFieldRef type is an abstract type.

Field Name	XML Schema Type	Req	Notes
internalId	xsd:string (attribute)	Y	References a unique instance of a custom field type.

To locate the internal ID for a specific custom field in the UI, go to Customization > Lists, Records, & Fields > *[Custom Field]* (where *[Custom Field]* is the type of custom field such as CRM). The internal IDs for each custom field that has been created is listed in the ID column in the UI.

LongCustomFieldRef

The LongCustomFieldRef type extends the CustomFieldRef abstract type.

Field Name	XML Schema Type	Req	Notes
value	xsd:int	Y	
internalId	xsd:string (attribute)	Y	References a unique instance of a custom field type.

DoubleCustomFieldRef

The DoubleCustomFieldRef type extends the CustomFieldRef abstract type.

Field Name	XML Schema Type	Req	Notes
value	xsd:double	Y	
internalId	xsd:string (attribute)	Y	References a unique instance of a custom field type.

BooleanCustomFieldRef

The BooleanCustomFieldRef type extends the CustomFieldRef abstract type.

Field Name	XML Schema Type	Req	Notes
value	xsd:boolean	Y	
internalId	xsd:string (attribute)	Y	References a unique instance of a custom field type.

StringCustomFieldRef

The StringCustomFieldRef type extends the CustomFieldRef abstract type.

Field Name	XML Schema Type	Req	Notes
value	xsd:string	Y	
internalId	xsd:string (attribute)	Y	References a unique instance of a custom field type.

DateCustomFieldRef

The DateCustomFieldRef type extends the CustomFieldRef abstract type.

Field Name	XML Schema Type	Req	Notes
value	xsd:datetime	Y	
internalId	xsd:string (attribute)	Y	References a unique instance of a custom field type.

SelectCustomFieldRef

The SelectCustomFieldRef type extends the CustomFieldRef abstract type. This references a single ListOrRecordRef and also requires an InternalId attribute to indicate the field name.

Field Name	XML Schema Type	Req	Notes
value	ListorRecordRef	Y	A single ListOrRecordRef.
internalId	xsd:string (attribute)	Y	References a unique instance of a custom field type.

MultiSelectCustomFieldRef

The MultiSelectCustomFieldRef type extends the CustomFieldRef abstract type. This references an array of ListOrRecordRef's and also requires an internalId attribute to indicate the field name.

Field Name	XML Schema Type	Req	Notes
value	ListorRecordRef[]	Y	An array of type RecordRef
internalId	xsd:string (attribute)	Y	References a unique instance of a custom field type.

CustomFieldList

Field Name	XML Schema Type	Req	Notes
value	customFieldRef []	Y	An array of type customFieldRef. The actual entries in the array will be of a concrete type that extends customFieldRef.

The following is an XML excerpt from a SOAP body that illustrates a custom field list that contains all the available custom field types.

SOAP Sample

```
<listRel:customFieldList xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"
>
 <platformCore:customField internalId="526" scriptId="custentity_exchange_rate" xsi:type="platf
ormCore:DoubleCustomFieldRef">
   <platformCore:value>2.35</platformCore:value>
 </platformCore:customField>
 <platformCore:customField internalId="524" scriptId="custentity_soft_skills" xsi:type="platfor
mCore:MultiSelectCustomFieldRef">
   <platformCore:value internalId="1">
      <platformCore:name>Communication</platformCore:name>
   </platformCore:value>
   <platformCore:value internalId="5">
      <platformCore:name>Teamwork</platformCore:name>
   </platformCore:value>
 </platformCore:customField>
 <platformCore:customField internalId="146" scriptId="custentity_checkbox" xsi:type="platformCo
re:BooleanCustomFieldRef">
   <platformCore:value>false</platformCore:value>
 </platformCore:customField>
 <platformCore:customField internalId="525" scriptId="custentity_first_contact" xsi:type="platf
ormCore:DateCustomFieldRef">
   <platformCore:value>2017-03-16T00:00:00.000-07:00</platformCore:value>
 </platformCore:customField>
 <platformCore:customField internalId="324" scriptId="custentity_proficiency_level" xsi:type="p
latformCore:SelectCustomFieldRef">
   <platformCore:value internalId="2">
      <platformCore:name>Medium</platformCore:name>
   </platformCore:value>
 </platformCore:customField>
 <platformCore:customField internalId="72" scriptId="custentity_franchiseeabn_on_customerrec" x
si:type="platformCore:StringCustomFieldRef">
   <platformCore:value>{partner.vatregnumber}</platformCore:value>
 </platformCore:customField>
 <platformCore:customField internalId="424" scriptId="custentity_test_score" xsi:type="platform
Core:LongCustomFieldRef">
   <platformCore:value>25</platformCore:value>
 </platformCore:customField>
</listRel:customFieldList>
```

Search Types

Every NetSuite record that supports search has corresponding search and advanced search objects. For example, the SuiteTalk listRel XSD contains a Customer object, as well as its corresponding CustomerSearch, CustomerSearchAdvanced, and CustomerSearchRow search objects.

- When using a < *Record* > **Search** object, all search fields within this search object belong to one of the types decribed in Search XML Schema Types.

- When searching on custom records or custom fields, all search fields belong to one of the types described in Search Custom Field XML Schema Types.
- When using either the < *Record* > **SearchAdvanced** or < *Record* > **SearchRow** search objects, all search fields within these “advanced search” objects belong to one of the types described in Search Column Custom XML Schema Types or Search Column Custom XML Schema Types.

Search XML Schema Types

The following sections define available search types. Every search field within a search object type belongs to one of these search types.

SearchPreferences

Field Name	XML Schema Type	Req
bodyFieldsOnly	boolean	Y/N
pageSize	int	Y/N
returnSearchColumns	boolean	Y/N

SearchRequest

Field Name	XML Schema Type	Req	Notes
preferences	boolean	Y	
searchRecord	SearchRecord	Y	

SearchResult

Field Name	XML Schema Type	Req	Notes
totalRecords	int	N	
pageSize	int	N	
totalPages	int	N	
pageIndex	int	N	
searchId	string	N	
status			
recordList	platformCore:RecordList		
searchRowList	platformCore:SearchRowList		

SearchStringField

Field Name	XML Schema Type	Req	Notes
operator	platformCoreTyp:	Y	Reference to a value in a system list. For more information on

Field Name	XML Schema Type	Req	Notes
	SearchStringFieldOperator (attribute)		available values, see Platform Enumerations.
searchValue	xsd:string	Y	

SearchBooleanField

Field Name	XML Schema Type	Req	Notes
operator	xsd:boolean	Y	The available values are true or false.

SearchDoubleField

Field Name	XML Schema Type	Req	Notes
operator	platformCoreTyp: SearchDoubleFieldOperator(attribute)	Y	Reference to a value in a system list. For more information on available values, see Platform Enumerations.
searchValue	xsd:double	Y	
searchValue2	xsd:double	N	If the operator is between or notBetween searchValue2 must be populated.

SearchLongField

Field Name	XML Schema Type	Req	Notes
operator	platformCoreTyp: SearchLongField Operator(attribute)	Y	Reference to a value in a system list. For more information on available values, see Platform Enumerations.
searchValue	xsd:long	Y	
searchValue2	xsd:long	N	If the operator is between or notBetween searchValue2 must be populated.

SearchTextNumberField

Field Name	XML Schema Type	Req	Notes
operator	platformCoreTyp: SearchTextNumberFieldOperator (attribute)	Y	Reference to a value in a system list. For more information on available values, see Platform Enumerations.

Field Name	XML Schema Type	Req	Notes
searchValue	xsd:string	Y	
searchValue2	xsd:string	N	If the operator is between or notBetween searchValue2 must be populated.

SearchDateField

Field Name	XML Schema Type	Req	Notes
operator	platformCoreTyp: SearchDate (attribute)	Y	Reference to a value in a system list. For more information on available values, see Platform Enumerations.
predefinedSearchValue	plateformCoreTyp: SearchDate	N	Reference to a value in a system list. For more information on available values, see Platform Enumerations.
searchValue	xsd:dateTime	N	Either predefinedSearchValue or searchValue should be populated.
searchValue2	xsd: dateTime	N	If the operator is between or notBetween searchValue2 must be populated.

SearchMultiSelectField

This search type is used to specify a list of one or more internal IDs that reference other user defined records in the system.

Note: Note that the maximum number of values that can be specified in a MultiSelectField is 1000.

Field Name	XML Schema Type	Req	Notes
operator	platformCoreTyp: SearchMultiSelectFieldOperator (attribute)	Y	Reference to a value in a system list. For more information on available values, see Platform Enumerations.
searchValue	platformCore: RecordRef	Y	An array of type SearchMultiSelectRefValue.

SearchEnumMultiSelectField

This search type is used to specify a list of one or more system defined constants.

Field Name	XML Schema Type	Req	Notes
operator	platformCoreTyp: SearchEnumMultiSelectFieldOperator (attribute)	Y	Reference to a value in a system list. For more

Field Name	XML Schema Type	Req	Notes
			information on available values, see Platform Enumerations.
searchValue	xsd:string	Y	

Search Custom Field XML Schema Types

The following sections define the available search types for custom fields. Every search field within a search object type belongs to one of these search types.

SearchCustomField

This is an abstract type.

Field Name	XML Schema Type	Req	Notes
id	xsd:string (attribute)	Y	References a unique instance of a custom field

SearchStringCustomField

The SearchStringCustomField type extends the SearchCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
internalId	xsd:string (attribute)	Y	Inherited from the SearchCustomField. Reference a unique instance of a custom field.
operator	platformCoreTyp: SearchStringFieldOperator (attribute)	Y	The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see Platform Enumerations.
searchValue	xsd:string	Y	

SearchBooleanCustomField

The SearchBooleanCustomField type extends the SearchCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
internalId	xsd:string (attribute)	Y	Inherited from the SearchCustomField. Reference a unique instance of a custom field.
searchValue	xsd: boolean	Y	The type is an enumeration type that restricts the value to true or false.

SearchLongCustomField

The SearchLongCustomField type extends the SearchCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
internalId	xsd:string (attribute)	Y	Inherited from the SearchCustomField. Reference a unique instance of a custom field.
operator	platformCoreTyp: SearchLongFieldOperator (attribute)	Y	The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see Platform Enumerations.
searchValue	xsd:long	Y	
searchValue2	xsd:long	N	

SearchDoubleCustomField

The SearchDoubleCustomField type extends the SearchCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
internalId	xsd:string (attribute)	Y	Inherited from the SearchCustomField. Reference a unique instance of a custom field.
operator	platformCoreTyp: SearchDouble FieldOperator (attribute)	Y	The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see Platform Enumerations.
searchValue	xsd:double	Y	
searchValue2	xsd:double	N	

SearchDateCustomField

The SearchDateCustomField type extends the SearchCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
internalId	xsd:string (attribute)	Y	Inherited from the SearchCustomField. Reference a unique instance of a custom field.
operator	platformCoreTyp: SearchDateF ieldOperator (attribute)	Y	The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see Platform Enumerations.

Field Name	XML Schema Type	Req	Notes
predefinedSearchValue	platformCoreTyp: SearchDate		
searchValue	xsd:dateTime	Y	
searchValue2	xsd:dateTime	Y	

SearchMultiSelectCustomField

The SearchMultiSelectCustomField type extends the SearchCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
internalId	xsd:string (attribute)	Y	Inherited from the SearchCustomField. Reference a unique instance of a custom field.
operator	platformCoreTyp: SearchMultiSelectFieldOperator (attribute)	Y	The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see Platform Enumerations.
searchValue	ListOrRecordRef	Y	

SearchEnumMultiSelectCustomField

The SearchEnumMultiSelectCustomField type extends the SearchCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
internalId	xsd:string (attribute)	Y	Inherited from the SearchCustomField. Reference a unique instance of a custom field.
operator	platformCoreTyp: SearchEnum MultiSelectFieldOperator (attribute)	Y	The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see Platform Enumerations.
searchValue	xsd:string	Y	

SearchCustomFieldList

Field Name	XML Schema Type	Req	Notes
customField	platformCore: SearchCustomField []	Y	

Search Column XML Schema Types

The following search types support advanced search functionality in web services.

SearchColumnField

This is an abstract type.

Field Name	XML Schema Type	Req	Notes
customLabel	xsd:string		

SearchColumnBooleanField

The SearchColumnBooleanField type extends the SearchColumnField abstract type.

Field Name	XML Schema Type	Req	Notes
searchValue	xsd:boolean	Y	

SearchColumnStringField

The SearchColumnStringField type extends the SearchColumnField abstract type.

Field Name	XML Schema Type	Req	Notes
searchValue	xsd:string	Y	

SearchColumnLongField

The SearchColumnLongField type extends the SearchColumnField abstract type.

Field Name	XML Schema Type	Req	Notes
searchValue	xsd:long	Y	

SearchColumnTextNumberField

The SearchColumnStringField type extends the SearchColumnField abstract type.

Field Name	XML Schema Type	Req	Notes
searchValue	xsd:string	Y	

SearchColumnDoubleField

The SearchColumnDoubleField type extends the SearchColumnField abstract type.

Field Name	XML Schema Type	Req	Notes
searchValue	xsd:double	Y	

SearchColumnDateField

The SearchColumnDateField type extends the SearchColumnField abstract type.

Field Name	XML Schema Type	Req	Notes
searchValue	xsd:dateTime	Y	

SearchColumnEnumSelectField

The SearchColumnEnumSelectField type extends the SearchColumnField abstract type.

Field Name	XML Schema Type	Req	Notes
searchValue	xsd:string	Y	

SearchColumnSelectField

The SearchColumnSelectField type extends the SearchColumnField abstract type. This references a single ListOrRecordRef and also requires an internalId attribute to indicate the field name.

Field Name	XML Schema Type	Req	Notes
searchValue	RecordRef	Y	

Search Column Custom XML Schema Types

The following search types support advanced search functionality in web services.

SearchColumnCustomField

This is an abstract type.

Field Name	XML Schema Type	Req	Notes
customLabel	xsd:string		

SearchColumnBooleanCustomField

The SearchColumnBooleanField type extends the SearchColumnCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
searchValue	xsd:boolean	Y	
internalId	xsd:string (attribute)	Y	

SearchColumnStringCustomField

The SearchColumnStringCustomField type extends the SearchColumnCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
searchValue	xsd:string	Y	
internalId	xsd:string (attribute)	Y	

SearchColumnLongCustomField

The SearchColumnLongCustomField type extends the SearchColumnCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
searchValue	xsd:long	Y	
internalId	xsd:string (attribute)	Y	

SearchColumnDoubleCustomField

The SearchColumnDoubleCustomField type extends the SearchColumnCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
searchValue	xsd:double	Y	
internalId	xsd:string (attribute)	Y	

SearchColumnDateCustomField

The SearchColumnDateCustomField type extends the SearchColumnCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
searchValue	xsd:dateTime	Y	
internalId	xsd:string (attribute)	Y	

SearchColumnEnumMultiSelectCustomField

The SearchColumnEnumMultiSelectCustomField type extends the SearchColumnCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
searchValue	xsd:string	Y	
internalId	xsd:string (attribute)	Y	

SearchColumnSelectCustomField

The SearchColumnSelectCustomField type extends the SearchColumnCustomField abstract type. This references a single ListOrRecordRef and also requires an internalId attribute to indicate the field name.

Field Name	XML Schema Type	Req	Notes
searchValue	ListOrRecordRef	Y	A single ListOrRecordRef
internalId	xsd:string (attribute)	Y	

SearchColumnMultiSelectCustomField

The SearchColumnMultiSelectCustomField type extends the SearchColumnCustomField abstract type. This references an array of ListOrRecordRefs.

Field Name	XML Schema Type	Req	Notes
searchValue	ListOrRecordRef[]	Y	
internalId	xsd:string (attribute)	Y	

SearchColumnCustomFieldList

The SearchColumnCustomFieldList type extends the SearchColumnCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
customField	SearchColumnCustomField		

Sample Code

SOAP Request — Opportunity Search

Following is an example that contains an excerpt of the SOAP body that illustrates an opportunity search containing several search field types.

```
<opportunitySearch>
      <projectedTotal operator="lessThan">
         <searchValue>100000</searchValue>
      </projectedTotal>
      <title operator="contains">
         <searchValue>Enterprise</searchValue>
      </title>
      <createdDateRange operator="between">
         <fromValue>2003-10-02</fromValue>
         <toValue>2003-10-12</toValue>
      </createdDateRange>
      <opportunityStatusList operator="anyOf">
         <searchValue>inProgress</searchValue>
         <searchValue>closedWon</searchValue>
      </opportunityStatusList>
      <customFieldList>
         <customField xsi:type="SearchSelectCustomField" internalId="35" scriptId="cust_SalesEn
gineer"
         operator="equals">
            <searchValue>Buddy Williams</searchValue>
         </customField>
         <customField xsi:type="SearchBooleanCustomField" internalId="165" scriptId="cust_hasSa
lesEngineer"
         operator="true"/>
         <customField xsi:type="SearchStringCustomField" internalId="322" scriptId="cust_DemoNo
tes"
         operator="startsWith">
            <searchValue>CRM</searchValue>
         </customField>
         <customField xsi:type="SearchMultiSelectCustomField" internalId="155" scriptId="cust_P
roductAreas"
         operator="noneOf">
            <searchValue>Inventory</searchValue>
            <searchValue>Warehousing</searchValue>
         </customField><
      </customFieldList>
</opportunitySearch>
```

Platform Enumerations

The following search types are used throughout web services to populate system defined lists. The tables below outline the available values that should be used to populate these fields in your web services requests. These enumerations are defined in the platformCoreTyp XSD.

Search Types (Table 1)

Enumerations	SearchString FieldOperator	SearchLongFieldOperator	SearchDoubleFieldOperator	SearchPeriod Field	SearchTime FieldOperator
is	X				
isNot	X				
startsWith	X				
doesNotStartWith	X				
contains	X				
doesNotContain	X				
equalTo		X	X	X	
lessThan		X	X		
greaterThan		X	X		
lessThanOrEqualTo		X	X		
greaterThanOrEqualTo		X	X		
notEqualTo		X	X		
notLessThan		X	X		
notGreaterThan		X	X		
notLessThanOrEqualTo		X	X		
notGreaterThanOrEqualTo		X	X		
between		X	X		X
notBetween		X	X		
empty		X	X		

Search Types (Table 1)

Enumerations	SearchString FieldOperator	SearchLongFieldOperator	SearchDoubleFieldOperator	SearchPeriod Field	SearchTime FieldOperator
notEmpty		X	X		

Search Types (Table 2)

Enumerations	SearchDate FieldOperator	SearchEnum MultiSelect FieldOperator	SearchMulti SelectField Operator	SearchDate
anyOf		X	X	
noneOf		X	X	
on	X			
before	X			
after	X			
onOrBefore	X			
onOrAfter	X			
within	X			
empty	X			
notOn	X			
notBefore	X			
notAfter	X			
notOnOrBefore	X			
notOnOrAfter	X			
notWithin	X			
notEmpty	X			
fiscalHalfBeforeLast				X
fiscalHalfBeforeLastToDate				X
fiscalQuarterBeforeLast				X
fiscalQuarterBeforeLastToDate				X
fiscalYearBeforeLast				X
fiscalYearBeforeLastToDate				X
fiveDaysAgo				X
fiveDaysFromNow				X
fourDaysAgo				X
fourDaysFromNow				X
fourWeeksStartingThisWeek				X
lastBusinessWeek				X
lastFiscalHalf				X
lastFiscalHalfOneFiscalYearAgo				X
lastFiscalHalfToDate				X
lastFiscalQuarter				X
lastFiscalQuarterOneFiscalYearAgo				X
lastFiscalQuarterToDate				X
lastFiscalQuarterTwoFiscalYearsAgo				X

Search Types (Table 2)

Enumerations	SearchDate FieldOperator	SearchEnum MultiSelect FieldOperator	SearchMulti SelectField Operator	SearchDate
lastFiscalYear				X
lastFiscalYearToDate				X
lastMonth				X
lastMonthOneFiscalQuarterAgo				X
lastMonthOneFiscalYearAgo				X
lastMonthToDate				X
lastMonthTwoFiscalQuartersAgo				X
lastMonthTwoFiscalYearsAgo				X
lastRollingHalf				X
lastRollingQuarter				X
lastRollingYear				X
lastWeek				X
lastWeekToDate				X
monthAfterNext				X
monthAfterNextToDate				X
monthBeforeLast				X
monthBeforeLastToDate				X
nextBusinessWeek				X
nextFiscalHalf				X
nextFiscalQuarter				X
nextFiscalYear				X
nextFourWeeks				X
nextMonth				X
nextOneHalf				X
nextOneMonth				X
nextOneQuarter				X
nextOneWeek				X
nextOneYear				X
nextWeek				X
ninetyDaysAgo				X
ninetyDaysFromNow				X
oneYearBeforeLast				X
previousFiscalQuartersLastFiscalYear				X
previousFiscalQuartersThisFiscalYear				X
previousMonthsLastFiscalHalf				X
previousMonthsLastFiscalQuarter				X
previousMonthsLastFiscalYear				X

Search Types (Table 2)

Enumerations	SearchDate FieldOperator	SearchEnum MultiSelect FieldOperator	SearchMulti SelectField Operator	SearchDate
previousMonthsSameFiscalHalfLastFiscalYear				X
previousMonthsSameFiscalQuarterLastFiscalYear				X
previousMonthsThisFiscalHalf				X
previousMonthsThisFiscalQuarter				X
previousMonthsThisFiscalYear				X
previousOneDay				X
previousOneHalf				X
previousOneMonth				X
previousOneQuarter				X
previousOneWeek				X
previousOneYear				X
previousRollingHalf				X
previousRollingQuarter				X
previousRollingYear				X
sameDayFiscalQuarterBeforeLast				X
sameDayFiscalYearBeforeLast				X
sameDayLastFiscalQuarter				X
sameDayLastFiscalYear				X
sameDayLastMonth				X
sameDayLastWeek				X
sameDayMonthBeforeLast				X
sameDayWeekBeforeLast				X
sameFiscalHalfLastFiscalYear				X
sameFiscalHalfLastFiscalYearToDate				X
sameFiscalQuarterFiscalYearBeforeLast				X
sameFiscalQuarterLastFiscalYear				X
sameFiscalQuarterLastFiscalYearToDate				X
sameMonthFiscalQuarterBeforeLast				X
sameMonthFiscalYearBeforeLast				X
sameMonthLastFiscalQuarter				X
sameMonthLastFiscalQuarterToDate				X

Search Types (Table 2)

Enumerations	SearchDate FieldOperator	SearchEnum MultiSelect FieldOperator	SearchMulti SelectField Operator	SearchDate
sameMonthLastFiscalYear				X
sameMonthLastFiscalYearToDate				X
sameWeekFiscalYearBeforeLast				X
sameWeekLastFiscalYear				X
sixtyDaysAgo				X
sixtyDaysFromNow				X
startOfFiscalHalfBeforeLast				X
startOfFiscalQuarterBeforeLast				X
startOfFiscalYearBeforeLast				X
startOfLastBusinessWeek				X
startOfLastFiscalHalf				X
startOfLastFiscalHalfOneFiscalYearAgo				X
startOfLastFiscalQuarter				X
startOfLastFiscalQuarterOneFiscalYearAgo				X
startOfLastFiscalYear				X
startOfLastMonth				X
startOfLastMonthOneFiscalQuarterAgo				X
startOfLastMonthOneFiscalYearAgo				X
startOfLastRollingHalf				X
startOfLastRollingQuarter				X
startOfLastRollingYear				X
startOfLastWeek				X
startOfMonthBeforeLast				X
startOfNextBusinessWeek				X
startOfNextFiscalHalf				X
startOfNextFiscalQuarter				X
startOfNextFiscalYear				X
startOfNextMonth				X
startOfNextWeek				X
startOfPreviousRollingHalf				X
startOfPreviousRollingQuarter				X
startOfPreviousRollingYear				X

Search Types (Table 2)

Enumerations	SearchDate FieldOperator	SearchEnum MultiSelect FieldOperator	SearchMulti SelectField Operator	SearchDate
startOfSameFiscalHalfLastFiscalYear				X
startOfSameFiscalQuarterLastFiscalYear				X
startOfSameMonthLastFiscalQuarter				X
startOfSameMonthLastFiscalYear				X
startOfThisBusinessWeek				X
startOfThisFiscalHalf				X
startOfThisFiscalQuarter				X
startOfThisFiscalYear				X
startOfThisMonth				X
startOfThisWeek				X
startOfThisYear				X
startOfWeekBeforeLast				X
tenDaysAgo				X
tenDaysFromNow				X
thirtyDaysAgo				X
thirtyDaysFromNow				X
thisBusinessWeek				X
thisFiscalHalf				X
thisFiscalHalfToDate				X
thisFiscalQuarter				X
thisFiscalQuarterToDate				X
thisFiscalYear				X
thisFiscalYearToDate				X
thisMonth				X
thisMonthToDate				X
thisRollingHalf				X
thisRollingQuarter				X
thisRollingYear				X
thisWeek				X
thisWeekToDate				X
thisYear				X
threeDaysAgo				X
threeDaysFromNow				X
threeFiscalQuartersAgo				X
threeFiscalQuartersAgoToDate				X
threeFiscalYearsAgo				X
threeFiscalYearsAgoToDate				X
threeMonthsAgo				X
threeMonthsAgoToDate				X

Search Types (Table 2)

Enumerations	SearchDate FieldOperator	SearchEnum MultiSelect FieldOperator	SearchMulti SelectField Operator	SearchDate
today				X
tomorrow				X
twoDaysAgo				X
twoDaysFromNow				X
weekAfterNext				X
weekAfterNextToDate				X
weekBeforeLast				X
weekBeforeLastToDate				X
yesterday				X

Important: The onOrAfter, onOrBefore, notOnOrAfter and notOnOrBefore operators only consider the date you specify in your request, and ignore the time you specify. If you need to specify not only the date but also the time in your search, use the after or before operators. Note, however, when you use these operators, the exact time specified in your request is not part of the result set.

Web Services Operations

SuiteTalk exposes NetSuite as a data source for programmatic access, hence most of the data operations developers expect, such as insert, update, delete, and select/search are supported. There are also a number of operations that are available as supporting operations for data read/write operations (for example: initialize and getSelectValue), or for providing metadata (for example: getCustomizationId), or for exposing an application function to programmatic access (for example: attach, detach).

Note: If an operation you are looking for doesn't appear in the list of operations below, look at the Support for Existing WSDL Versions for your version of the WSDL.

See these sections for complete details on all SuiteTalk operations:

- Web Services Standard Operations
- Web Services List Operations
- Web Services Asynchronous Operations

Important: NetSuite hosts customer accounts in multiple data centers. For that reason, the correct URL for web services access varies depending on the data center hosting the account. Your integration **must** incorporate logic that dynamically determines the correct URL. With the 2012.2 and later endpoints, you should use the getDataCenterUrls operation to dynamically discover the correct URL. With older endpoints, you should use the REST roles service. For details, see the help topic Using the REST roles Service to Get User Accounts, Roles, and Domains. Starting from version 2017.2, you can also use the DataCenterUrls REST Service.

Web Services Standard Operations

The following operations are supported in SuiteTalk. They are organized in alphabetical order:

Operation / API	**Summary**
add / addList	Use to add one or more records into the system. The system returns a NetSuite identifier (internalId) that is unique for each record created within a record type.
attach / detach	Use to attach or detach another record or file to/from another record.
changeEmail	Use to change the email address for the account.
changePassword	Use to change the password for the account.
changePasswordOrEmail	Use to change a users email or password.
checkAsyncStatus	Use to check the status of an asynchronous web services submission.
delete / deleteList	Use to delete one or more records in the system. The records to be deleted are identified by either the internal or external ID and the record type.
get / getList	Use to query the system for one or more records. You must provide either the internal or external ID and the record type for each query item.
getAll	Use to return a list of records that do not have a search interface.

Operation / API	Summary
getAsyncResult	Use to retrieve the results of an asynchronous web services submission.
getBudgetExchangeRate	Use to get and filter all data related to the Budget Exchange Rates table.
getConsolidatedExchangeRate	Use to get and filter all data related to the Consolidated Exchange Rates table. **Important:** The exposure of the consolidated exchange rate record prompted the removal of the getConsolidatedExchangeRate operation from the 2017.1 endpoint. You can still use the getConsolidatedExchangeRate operation on earlier endpoints. For more information about working with consolidated exchange rates on endpoints starting from 2017.1, see the help topic Consolidated Exchange Rate.
getCurrencyRate	Use to get currency exchange rates from the Currency Exchange Rate table.
getCustomizationId	Use to retrieve the internalIds, externalIds, and/or scriptIds of all custom objects of a specified type.
getDataCenterUrls	Use for dynamic discovery of the correct URL for web services requests. **Important:** All web services integrations **must** include logic to dynamically discover the correct URL. If the getDataCenterURLs operation is not included in the endpoint you are using, use the REST roles service to obtain the correct URL. For details on this service, see the help topic Using the REST roles Service to Get User Accounts, Roles, and Domains.
getDeleted	Use to retrieve a list of deleted records of a particular type during a specified period.
getItemAvailability	Use to retrieve the inventory availability for a specific list of items.
getPostingTransactionSummary	Use to retrieve a summary of the actual data in an account.
getSavedSearch	Use to retrieve a list of existing saved searches for each record type. Specify the search record type to get a list of record references of the saved search.
getSelectValue	Use to retrieve valid values for a particular recordRef field where the referenced record type is not yet exposed in the web services API or when the logged in role does not have permission to the instances of the record type.
getServerTime	Use to get server time, resulting in more accurate and reliable sync'ing of data than using using local client time. The client will use the time from server to determine if the record has changed since the last synchronization.
initialize / initializeList	Use to emulate the UI workflow by pre-populating fields on transaction line items with values from a related record. The initializeList operation can be used to run batch processes to retrieve initialized records.

Operation / API	Summary
login	Use to login into NetSuite. This operation is similar to the NetSuite UI and requires you to provide a valid username, password, role, and account number.
logout	Use to logout from the system. The logout operation invalidates the current session.
mapSso	Use to automate the mapping between external applications credentials and NetSuite credentials for a user.
search	Use to search for a set of records based on specific search criteria. This operation supports pagination, so that large result sets can be retrieved in smaller sets. For more information on how to control pagination, refer to Web Services Preferences.
searchMore	Used to retrieve more records after an initial search operation is invoked.
searchMoreWithId	Use to retrieve search results for users who send requests to NetSuite by providing request-level credentials, rather than by invoking login. In this case, users cannot call searchMore or searchNext.
searchNext	Use to retrieve the next set of records after an initial search operation is invoked.
ssoLogin	Use to establish a single sign-on connection. This operation allows for a partner application to login on behalf of the user in to NetSuite, without the user's credentials ever going through partner servers.
update / updateList	Use to update one or more existing records in the system by providing new values for the fields to be updated for each record. The records to be updated are identified by either the internal or external ID and the record type.
updateInviteeStatus / updateInviteeStatusList	Allows event invitees to accept or decline NetSuite events. After invitees have responded to the event, the Event record is updated with their response.
upsert / upsertList	Use to add new records and update existing records in a single operation. Records are identified by external ID and record type. If a record of the specified type with a matching external ID exists in the system, it is updated. If it does not exist, a new record is created.

Web Services List Operations

Within a single soap request, only one operation can be performed. For example, one add, one addList or one delete. However, a single **list** operation (addList, updateList, upsertList, deleteList, getList, and initializeList) lets you work with multiple record types. For example, with a single addList operation you can add 3 customers, 4 opportunities, and 1 contact.

Note that list operations process records in the order in which the records are submitted. For example, an addList operation will add all records in the order they appear in the list.

Web Services Asynchronous Operations

The following asynchronous equivalents are available for these list operations:

- addList: asyncAddList
- updateList: asyncUpdateList
- upsertList: asyncUpsertList
- deleteList: asyncDeleteList
- getList: asyncGetList
- search: asyncSearch
- initializeList: asyncInitializeList

You can use the checkAsyncStatus and getAsyncResult operations to track asynchronous requests. For information on asynchronous request processing, see Synchronous Versus Asynchronous Request Processing.

add

The add operation is used to add a new instance of a record in NetSuite. It is similar to the addList operation except that it allows only one record to be added at a time. Note that to prevent duplicate records, NetSuite recommends using the alternate upsert and upsertList operations along with external ids to add records to NetSuite.

Note: Although records of a particular type may be used in multiple integration scenarios, each record instance can only have a single external ID value. To maintain data integrity, only a single integrated application can set and update external ID values for each record type. External ID values for all records of each type must all come from the same external application.

Request

The AddRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
record	Record	The record type is an abstract type so an instance of a type that extends record must be used — such as Customer or Event.

Response

The AddResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
response	WriteResponse	Contains details on the status of the operation and a reference to the created record.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault

- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In this example, a single customer record is added.

```
<soap:Body>
<platformMsgs:add>
        <platformMsgs:record xsi:type="listRel:Customer">
                <listRel:entityId>Shutter Fly</listRel:entityId>
                <listRel:companyName>Shutter Fly, Inc</listRel:companyName>
                <listRel:unsubscribe>false</listRel:unsubscribe>
        </platformMsgs:record>
</platformMsgs:add>
</soap:Body>
```

SOAP Response

In the response, notice that the internalID for the record added is returned.

```
<soapenv:Body>
<addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<writeResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"/>

      <baseRef internalId="979" type="customer" xsi:type="ns2:RecordRef"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
</writeResponse>
</addResponse>
</soapenv:Body>
```

C#

```
private void addCustomer()
{
   // This operation requires a valid session
   this.login( true );

   Customer customer = new Customer();

   // Set entityId, company name, and email
   if ( "true".Equals( _dataCollection["promptForFieldValues"] ) )
   {
      _out.writeLn(
      "\nPlease enter the following customer information. " +
      "Note that some fields have already been populated. " );
      _out.write( "Entity name: " );
      customer.entityId = _out.readLn();
      _out.write( "Company name: " );
      customer.companyName = _out.readLn();
      _out.write( "E-mail: " );
      customer.email = _out.readLn();
      _out.write( "Sales Rep key: " );
      RecordRef salesRep = new RecordRef();
      salesRep.internalId = _out.readLn();
      //salesRep.type = RecordType.contact;
      //salesRep.typeSpecified = true;
      customer.salesRep = salesRep;
   }
   else
   {
      customer.entityId = "XYZ Inc";
      customer.companyName = "XYZ, Inc.";
      customer.email = "bsanders@yahoo.com";
   }

   // Set email preference
   customer.emailPreference = CustomerEmailPreference._hTML;
   customer.emailPreferenceSpecified = true;

   // Set entity status. The nsKey can be obtained from Setup > SFA > Customer Statuses.
   // The default status is "Closed Won" which has an nsKey of 13
   RecordRef status = new RecordRef();
   if ( "true".Equals( _dataCollection["promptForFieldValues"] ) )
   {
      _out.write( "Entity status nsKey (press enter for default value of Closed Won): " );
      String statusKey = _out.readLn();
      if ( statusKey.Equals( "" ) )
      {
         status.internalId = "13";
      }
      else
      {
         status.internalId = statusKey;
      }
   }
   else
   {
      status.internalId = "13";
   }

   customer.entityStatus = status;

   // Populate the address list for this customer. You can put in as many
   // adresses as you like.
   CustomerAddressbook address = new CustomerAddressbook();
   address.defaultShipping = true;
   address.defaultShippingSpecified = true;
   address.defaultBilling = false;
   address.defaultBillingSpecified = true;
   address.label = "Shipping Address";
   address.addressee = "William Sanders";
   address.attention = "William Sanders";
   address.addr1 = "4765 Sunset Blvd";
   address.city = "San Francisco";
   address.state = "CA";
   address.zip = "94131";
   address.country = Country._unitedStates;

   // Attach the CustomerAddressbookList to the customer
   CustomerAddressbookList addressList = new CustomerAddressbookList();
   CustomerAddressbook[] addresses = new CustomerAddressbook[1];
   addresses[0] = address;
   addressList.addressbook = addresses;
   customer.addressbookList = addressList;

   // Invoke add() operation
   WriteResponse response = _service.add( customer );

   // Print the document id from the SOAP header
   //_out.info(
   //"\nThe add() operation with document id " + _service.documentInfo.nsID + " was processed "
 );

   // Process the response
   if ( response.status.isSuccess )
   {
      _out.info(
      "\nThe following customer was added successfully:" +
      "\nkey=" + ((RecordRef) response.baseRef).internalId +
      "\nentityId=" + customer.entityId +
      "\ncompanyName=" + customer.companyName +
      "\nemail=" + customer.email +
      "\nstatusKey=" + customer.entityStatus.internalId +
      "\naddressbookList[0].label=" + customer.addressbookList.addressbook[0].label );

      // Create a second customer that's a sub-customer of the first customer
      /*
      Customer subCustomer = new Customer();

      subCustomer.entityId = "XYZ Japan";

      // Set the parent customer
      recordRef = new RecordRef();
      recordRef.internalId = ((RecordRef) response.baseRef).internalId;
      //recordRef.internalId = "125";
      subCustomer.parent = recordRef;

      // Invoke add() operation
      response = _service.add( subCustomer );

      // Process the response
      //"\nThe add() operation with job id " + _service.sessionInfo.jobID + " was processed " )
;
      if ( !response.status.isSuccess )
```

```
            {
               _out.info( "\nThe customer was not added:" );
               _out.error( getStatusDetails( response.status ) );
            }
            else
            {
               _out.info(
               "\nThe following customer was added successfully:" +
               "\nkey=" + ((RecordRef) response.baseRef).internalId +
               "\nentityId=" + customer.entityId +
               "\ncompanyName=" + customer.companyName);
            }
            */
         }
         else
         {
            _out.error( "The customer was not added:", true );
            _out.error( getStatusDetails( response.status ) );
         }
      }
```

Java

```
public void addCustomer() throws RemoteException, ExceededRecordCountFault,
ExceededUsageLimitFault, InsufficientPermissionFault, InvalidSessionFault {
   // This operation requires a valid session
   this.login(true);

   Customer customer = new Customer();

   // Set entityId, company name, and email
   if ("true".equals(_properties.getProperty("promptForFieldValues"))) {
      _console
      .writeLn("\nPlease enter the following customer information. "
      + "Note that some fields have already been populated. ");
      _console.write("Entity name: ");
      customer.setEntityId(_console.readLn());
      _console.write("Company name: ");
      customer.setCompanyName(_console.readLn());
      _console.write("E-mail: ");
      customer.setEmail(_console.readLn());
   } else {
      customer.setEntityId("XYZ Inc");
      customer.setCompanyName("XYZ, Inc.");
      customer.setEmail("bsanders@yahoo.com");
   }

   // Set email preference.
   customer.setEmailPreference(CustomerEmailPreference._hTML);

   // Set entity status. The nsKey can be obtained from Setup > SFA >
   // Customer Statuses.
   // The default status is "Closed Won" which has an nsKey of 13
   RecordRef status = new RecordRef();
```

```
    if ("true".equals(_properties.getProperty("promptForFieldValues"))) {
       _console.write("Entity status nsKey (press enter for default value of Closed Won): ");
       String statusKey = _console.readLn();
       if (statusKey.equals("")) {
          status.setInternalId("13");
       } else {
          status.setInternalId(statusKey);
       }
    } else {
       status.setInternalId("13");
    }

    customer.setEntityStatus(status);

    // Populate the address list for this customer. You can put in as many
    // adresses as you like.
    CustomerAddressbook address = new CustomerAddressbook();
    address.setDefaultShipping(Boolean.TRUE);
    address.setDefaultBilling(Boolean.FALSE);
    address.setLabel("Shipping Address");
    address.setAddressee("William Sanders");
    address.setAttention("William Sanders");
    address.setAddr1("4765 Sunset Blvd");
    address.setCity("San Francisco");
    address.setState(“CA”);
    address.setZip("94131");
    address.setCountry(Country._unitedStates);

    // Attach the CustomerAddressbookList to the customer
    CustomerAddressbookList addressList = new CustomerAddressbookList();
    CustomerAddressbook[] addresses = new CustomerAddressbook[1];
    addresses[0] = address;
    addressList.setAddressbook(addresses);
    customer.setAddressbookList(addressList);

    // Invoke add() operation
    WriteResponse response = _port.add(customer);

    // Print the document id from the SOAP header
    // _console.info(
    // "\nThe add() operation with document id " + _port.documentInfo.nsID +
    // " was processed " );

    // Process the response
    if (response.getStatus().isIsSuccess()) {
       _console.info("\nThe following customer was added successfully:"
       + "\nkey="
       + ((RecordRef) response.getBaseRef()).getInternalId()
       + "\nentityId="
       + customer.getEntityId()
       + "\ncompanyName="
       + customer.getCompanyName()
       + "\nemail="
       + customer.getEmail()
       + "\nstatusKey="
```

```
        + customer.getEntityStatus().getInternalId()
        + "\naddressbookList[0].label="
        + customer.getAddressbookList().getAddressbook(0)
        .getLabel());
    } else {
        _console.error("The customer was not added:", true);
        _console.error(getStatusDetails(response.getStatus()));
    }
}
```

addList

The addList operation is used to add one or more new instances of a record to NetSuite. If there are multiple records, they can either be of the same record type or different record types. For example, it's possible to add a customer and a contact within a single request using this operation. However, each record entered must have a unique signature. Adding two records with the same signature results in a SOAP fault. The signature consists of parameters required to identify a record as unique.

For example, in the case of entities, a record is uniquely identified by its name, its type and its parent hierarchy. So you could have two records with the same entityId (or name) belonging to two different record types as follows:

Customer (the type):
John Smith
MyCompany: John Smith

Contact
John Smith

But a second record such as the following would be invalid:

Contact
John Smith

Note: An asynchronous equivalent is available for this operation, **asyncAddList**. For information about asynchronous request processing, see Synchronous Versus Asynchronous Request Processing.

Note that to prevent duplicate records, NetSuite recommends using the alternate upsert and upsertList operations when adding records to NetSuite.

Request

The AddListRequest type is used for the request.

Element Name	XSD Type	Notes
record[]	Record	Contains an array of record objects. The record type is an abstract type so an instance of a type that extends record must be used—such as Customer or Event.

Response

The AddListResponse type is used for the response.

Element Name	XSD Type	Notes
response[]	WriteResponse	Contains an array of WriteResponse objects, each of which contains details on the status of that add operation and a reference to that created record.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In the following example, two customer records are added. When using the addList operation, you can submit two different record types in the same request.

```
<soap:Body>
<platformMsgs:addList>
        <platformMsgs:record xsi:type="listRel:Customer">
                <listRel:entityId>Shutter Fly</listRel:entityId>
                <listRel:companyName>Shutter Fly, Inc</listRel:companyName>
                <listRel:unsubscribe>false</listRel:unsubscribe>
        </platformMsgs:record>
        <platformMsgs:record xsi:type="listRel:Customer">
                <listRel:entityId>GNC</listRel:entityId>
                <listRel:companyName>GNC Corp</listRel:companyName>
                <listRel:unsubscribe>false</listRel:unsubscribe>
        </platformMsgs:record>
</platformMsgs:addList>
</soap:Body>
```

SOAP Response

In the response, notice that the internalID for each record added is returned.

```
<soapenv:Body>
<addListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<writeResponseList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <writeResponse>
      <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com
```

```
"/>
         <baseRef internalId="980" type="customer" xsi:type="ns2:RecordRef"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
   </writeResponse>
   <writeResponse>
      <ns3:status isSuccess="true" xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com
"/>
         <baseRef internalId="981" type="customer" xsi:type="ns4:RecordRef"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com"/>
   </writeResponse>
</writeResponseList>
</addListResponse>
</soapenv:Body>
```

Java Sample

```
public void addListCustomer() throws RemoteException {
                this.login(true);

                Customer cust1 = new Customer();
                cust1.setEntityId("Shutter Fly");
                cust1.setCompanyName("Shutter Fly, Inc");
                cust1.setUnsubscribe(false);

                Customer cust2 = new Customer();
                cust2.setEntityId("GNC");
                cust2.setCompanyName("GNC Corp");
                cust2.setUnsubscribe(false);

                Customer[] customers = new Customer[2];
                customers[0] = cust1;
                customers[1] = cust2;

                WriteResponseList responseList = _port.addList(customers);
        }
```

attach / detach

The attach and detach operations can be used to define or remove a relationship between two records. For example, a Contact record can be associated with a Partner record, or an Opportunity record can be associated with a Customer record.

Important: A user error is thrown if you attempt to attach files or record that do not exist.

You can also use the attach / detach operations to attach or detach a file to or from a record. Any file that is in the NetSuite file cabinet, for example an MS Word or Excel file or a PDF can be attached to any record other than a custom record.

Note that when attaching Contacts to other entity records, the Contact's role can also be specified during the request. Contact Roles are roles available in a user defined list at List > Relationships > Contacts. This list has been exposed as ContactRole in accounting.xsd.

Note that to prevent duplicate records, NetSuite recommends using the alternate upsert and upsertList operations along with external ids to add records to NetSuite.

Note: Contact records can be attached to all entity records expect for other Contact or Group records.

The following table lists all records that support the attach/detach operations. It also lists which records can accept file attachments as well as which records can accept Contact records as attachments.

Record Type	Accepts File Attachments	Accepts Contact Record Attachments
Transactions		
Check	X	X
Custom Transaction	X	
Expense Report	X	
Inventory Adjustment	X	X
Item Fulfillment	X	X
Journal Entry	X	X
Intercompany Journal Entry	X	X
Opportunity	X	X
Sales Order	X	X
Customer Payment	X	X
Return Authorization	X	X
Credit Memo	X	X
Cash Refund	X	X
Estimate	X	X
Invoice	X	X
Cash Sale	X	X
Purchase Order	X	X
Purchase Order Receipt	X	X
Purchase Requisition	X	X
Customer Payment	X	X
Customer Refund	X	X
Customer Deposit	X	X
Customer Deposit Application	X	X
Vendor Bill	X	X
Work Order	X	
Work Order Issue	X	
Work Order Close	X	
Work Order Completion	X	

Record Type	Accepts File Attachments	Accepts Contact Record Attachments
Entities		
Customer	X	X
Contact	X	
Employee	X	
Partner	X	X
Vendor	X	X
Project	X	X
Group	X	**Note:** Although you cannot currently attach Contact records to Group, you can attach members to create a static group.
Activities		
Task	X	
Event	X	
Phone Call	X	
Project Task	X	
Support		
SupportCase	X	
Issue	X	
Marketing		
Campaign	X	

Request (attach)

The AttachRequest type is used for this request. It contains the following fields:

Element Name	XSD Type	Notes
attachReference	AttachReference	

Request (detach)

The DetachRequest type is used for this request. It contains the following fields:

Element Name	XSD Type	Notes
detachReference	DetachReference	

Response (attach)

The AttachResponse type is used for this response. It contains the following fields:

Element Name	XSD Type	Notes
response	WriteResponse	

Response (detach)

The DetachResponse type is used for this response. It contains the following fields:

Element Name	XSD Type	Notes
response	WriteResponse	

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request (attach)

```
<attach xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <attachReferece xsi:type="ns1:AttachContactReference"
      xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com">
      <ns1:attachTo internalId="176" type="customer" xsi:type="ns1:RecordRef">
        <ns1:name xsi:type="xsd:string">Adelina Shonkwiler</ns1:name>
          </ns1:attachTo>
       <ns1:contact internalId="1467" xsi:type="ns1:RecordRef"/>
         <ns1:contactRole internalId="-10" xsi:type="ns1:RecordRef">
            <ns1:name xsi:type="xsd:string">Primary Contact</ns1:name>
          </ns1:contactRole>
   </attachReferece>
 </attach>
```

SOAP Request (detach)

```
<detach xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
```

```
   <attachReferece xsi:type="ns1:AttachBasicReference"
         xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com">
          <ns1:attachTo internalId="176" type="customer" xsi:type="ns1:RecordRef">
           <ns1:name xsi:type="xsd:string">Adelina Shonkwiler</ns1:name>
               </ns1:attachTo>
   <ns1:attachedRecord internalId="1467" type="contact" xsi:type="ns1:RecordRef"/>
     </attachReferece>
 </detach>
```

SOAP Response (attach)

```
<attachResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <writeResponse>
      <ns1:status isSuccess="true"
            xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"/>
           <baseRef internalId="176" type="customer" xsi:type="ns2:RecordRef"
                  xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">
               <ns2:name>Adelina Shonkwiler</ns2:name>
            </baseRef>
     </writeResponse>
 </attachResponse>
```

SOAP Response (detach)

```
<detachResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <writeResponse>
     <ns1:status isSuccess="true"
         xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"/>
        <baseRef internalId="176" type="customer" xsi:type="ns2:RecordRef"
               xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">
             <ns2:name>Adelina Shonkwiler</ns2:name>
          </baseRef>
   </writeResponse>
</detachResponse>
```

Java (attach operation)

```
public void attach() throws Exception
    {
        RecordRef contactRef = new RecordRef();
        contactRef.setInternalId("1467");
        contactRef.setType(RecordType.contact);

        RecordRef contactRoleRef = new RecordRef();
        contactRoleRef.setInternalId("-10");
        contactRoleRef.setName("Primary Contact");

        RecordRef customerRef = new RecordRef();
        customerRef.setInternalId("176");
        customerRef.setType(RecordType.customer);
        AttachContactReference attachRef = new AttachContactReference();
        attachRef.setContact(contactRef);
        attachRef.setAttachTo(customerRef);
        attachRef.setContactRole(contactRoleRef);

        WriteResponse attachResponse = sessMgr.getPort().attach(attachRef);
    }
```

Java (detach operation)

```
public void detach() throws Exception
    {
        RecordRef contactRef = new RecordRef();
        contactRef.setInternalId("1467");
        contactRef.setType(RecordType.contact);

        RecordRef customerRef = new RecordRef();
        customerRef.setInternalId("176");
        customerRef.setType(RecordType.customer);

        AttachContactReference detachRef = new AttachContactReference();
        detachRef.setContact(contactRef);
        detachRef.setAttachTo(customerRef);

        WriteResponse detachResponse = sessMgr.getPort().detach(detachRef);
    }
```

changePasswordOrEmail

The changePasswordOrEmail operation is supported in the 2012.1 and earlier endpoints. Use this operation to change **either** a user's email address or password. You cannot use this operation to change both the email address and password in a single call.

Warning: If you upgrade to the 2012.2 endpoint or a later version, any existing calls to changeEmailOrPassword will no longer work. Instead, you must use the changeEmail operation to change a user's email address, and you must use changePassword to change a user's password

Important: When using the changePasswordOrEmail operation to change email addresses, be aware that the addresses are not changed immediately. The old email address remains in effect until the new email address is validated by the user, through a validation link sent to the new address. A notification is also sent to the old address. This behavior is the same as when users change their email addresses through the user interface.

For details on how to change a user's email address or password using the NetSuite user interface, see the following topics:

- Your User Credentials
- Change Email Link

- Change Password Link

Request

The ChangePasswordOrEmailRequest type is used for the request.

Element Name	XSD Type	Notes
changePasswordOrEmailCredentials	ChangePasswordOrEmailCredentials	

The ChangePasswordOrEmailCredentials type takes the following fields:

- currentPassword
- newEmail
- newEmail2
- newPassword
- newPassword2
- justThisAccount

Response

The ChangePasswordOrEmailResponse type is used for the response. It does not contain any fields.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InsufficientPermissionFault
- InvalidAccountFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- UnexpectedErrorFault
- InvalidVersionFault

Example — Changing an Email Address

The following example shows how to use the changeEmailOrPassword operation to change a user's email address.

Java

```
public void test_ChangePwdOrEmail() throws Exception {
   sessMgr.loginWithEmail("oldEmail@ws.com");
   ChangePasswordOrEmailCredentials c = new ChangePasswordOrEmailCredentials();
   c.setCurrentPassword("xxxxxx");
   c.setNewEmail("newEmail@ws.com");
   c.setNewEmail2("newEmail@ws.com");
   sessMgr.getPort().changePasswordOrEmail(c);
   sessMgr.loginWithEmail("newEmail@ws.com");
}
```

SOAP Request

```
<changePasswordOrEmail xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <changePasswordOrEmailCredentials>
      <ns1:currentPassword xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com">xxxxxx
<   ns1:currentPassword>
      <ns2:newEmail xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">newEmail@ws.c
om<ns2:newEmail>
      <ns3:newEmail2 xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">newEmail@ws.
com<ns3:newEmail2>
   </changePasswordOrEmailCredentials>
</changePasswordOrEmail>
```

SOAP Response

```
<changePasswordOrEmailResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <sessionResponse>
      <ns1:status isSuccess="true"xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"
/>
   </sessionResponse>
</changePasswordOrEmailResponse>
```

Example — Changing a Password

The following example shows how to change a password. Note that with this approach, you have to declare values for NewEmail and NewEmail2 — but you should specify empty strings. These fields are ignored, but you must declare them, or the operation fails.

Java

```
public void changePassword() throws Exception
{
   ChangePasswordOrEmailCredentials newCredentials = new ChangePasswordOrEmailCredentials();
   newCredentials.setCurrentPassword("currentPwd ");
   newCredentials.setNewPassword("newPwd ");
   newCredentials.setNewPassword2("newPwd ");
   newCredentials.setNewEmail("");   // NewEmail is mandatory, and you must specify an empty st
ring for the operation to complete successfully
   newCredentials.setNewEmail2("");  // NewEmail2 is mandatory, and you must specify an empty s
tring for the operation to complete successfully

   SessionResponse response = sessMgr.getPort().changePasswordOrEmail(newCredentials);
}
```

SOAP Request

```
<changePasswordOrEmail xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <changePasswordOrEmailCredentials>
      <ns7:currentPassword xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">curren
tPwd</ns7:currentPassword>
      <ns8:newEmail xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com"></ns8:newEmai
l>
      <ns9:newEmail2 xmlns:ns9="urn:core_2017_1.platform.webservices.netsuite.com"></ns9:newEma
il2>
      <ns10:newPassword xmlns:ns10="urn:core_2017_1.platform.webservices.netsuite.com">newPwd</
ns10:newPassword>
      <ns11:newPassword2 xmlns:ns11="urn:core_2017_1.platform.webservices.netsuite.com">newPwd<
/ns11:newPassword2>
   </changePasswordOrEmailCredentials>
</changePasswordOrEmail>
```

changeEmail

Use the changeEmail operation to change a user's email address.

This topic provides details about how to change a user's email address with web services code. For details about how to make this change in the NetSuite user interface, see the help topics Change Email Link and Your User Credentials.

Important: This operation is supported as of the 2012.2 endpoint. The 2012.2 and later endpoints support changeEmail and changePassword operations that replace changePasswordOrEmail. Endpoints 2012.2 and later **do not support** the changePasswordOrEmail operation.

Important: Be aware that a user’s email address is not changed immediately. The old email address remains in effect until the new email address is validated by the user, through a validation link sent to the new address. A notification is also sent to the old address. This behavior is the same as when users change their email addresses through the user interface.

Request

The ChangeEmailRequest type is used for the request.

Element Name	XSD Type	Notes
changeEmail	ChangeEmail	

The ChangeEmail type takes the following fields:

- currentPassword
- newEmail
- newEmail2
- justThisAccount

Response

The ChangeEmailResponse type is used for the response. It does not contain any fields.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InsufficientPermissionFault
- InvalidAccountFault
- InvalidCredentialsFault
- InvalidVersionFault
- ExceededRequestLimitFault
- UnexpectedErrorFault
- UserError

Sample Code

SOAP Request

```
<changeEmail xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <changeEmail>
      <ns6:currentPassword xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.com">xxxxxx
x</   ns6:currentPassword>
      <ns7:newEmail xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">newEmail@test
er.com</ns7:newEmail>
      <ns8:newEmail2 xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com"> newEmail @t
ester.com</ns8:newEmail2>
      <ns9:justThisAccount xmlns:ns9="urn:core_2017_1.platform.webservices.netsuite.com">true</
ns9:justThisAccount>
   </changeEmail>
</changeEmail>
```

SOAP Response

```
<changeEmailResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
              <sessionResponse>
                     <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.
platform.webservices.netsuite.com"/>
      </sessionResponse>
</changeEmailResponse>
```

Java

```
public void test_ChangeEmail() throws Exception {
    sessMgr.loginWithEmail("oldEmail@ws.com");
    ChangeEmail c = new ChangeEmail();
    c.setCurrentPassword("xxxxxx");
    c.setNewEmail("newEmail@ws.com");
    c.setNewEmail2("newEmail@ws.com");
    c.setJustThisAccount(true);
    sessMgr.getPort().changeEmail(c);
    sessMgr.loginWithEmail("newEmail@ws.com");
}
```

changePassword

Use the changePassword operation to change a user's NetSuite password.

This topic provides details about how to change a user's password with web services code. For details about how to make this change in the NetSuite user interface, see the help topics Change Password Link and Your User Credentials.

Request

The ChangePasswordRequest type is used for the request.

Element Name	XSD Type	Notes
changePassword	ChangePassword	

The ChangePassword type takes the following fields:

- currentPassword
- newPassword
- newPassword2

Response

The ChangePasswordResponse type is used for the response. It does not contain any fields.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InsufficientPermissionFault
- InvalidAccountFault
- InvalidCredentialsFault
- InvalidVersionFault
- ExceededRequestLimitFault
- UnexpectedErrorFault
- UserError

Sample Code

SOAP Request

```
      <soapenv:Body>
         <changePassword xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <changePassword>
               <ns6:currentPassword xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.co
m">password1</ns6:currentPassword>
               <ns7:newPassword xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">p
assword2</ns7:newPassword>
               <ns8:newPassword2 xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com">
password2</ns8:newPassword2>
            </changePassword>
         </changePassword>
      </soapenv:Body>
```

SOAP Response

```
      <soapenv:Body>
         <changePasswordResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">

            <sessionResponse>
               <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
            </sessionResponse>
         </changePasswordResponse>
      </soapenv:Body>
```

Java

```
ChangePassword changePW = new ChangePassword();
changePW.setCurrentPassword("password1");
changePW.setNewPassword("password2");
changePW.setNewPassword2("password2");

c.getPort().changePassword(changePW);
```

checkAsyncStatus

See checkAsyncStatus in Synchronous Versus Asynchronous Request Processing for details.

delete

The delete operation is used to delete an instance of a record. It is similar to the deleteList operation, except that it permits only one record to be deleted per request.

Request

The DeleteRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
recordRef	RecordRef	Indentifies the record to be deleted.
deletionReason	DeletionReason	Identifies a deletion reason and deletion reason memo. This parameter supports the Use Deletion Reasons feature, which requires users to provide reasons when they delete transactions. However, even when this feature is not enabled, you must use this parameter with every delete request. (You can provide a value of `null` in such cases.) For more details, see Deletion Reason Usage Notes.

Deletion Reason Usage Notes

Note the following about the deletionReason parameter:

- The deletionReason complex type includes two fields: deletionReasonCode and deletionReasonMemo. The deletionReasonCode must identify a deletion reason that is listed at Setup > Accounting > Accounting Lists. If the Use Deletion Reasons feature is enabled and you use the deletionReasonCode to identify a code that does not exist, the request fails with an `INVALID_REF_KEY` error.
- Deletion reasons can be saved only for transactions. However, in web services, you must use the deletionReason parameter even when referencing other record types, and even when the Use Deletion Reasons feature is not enabled. For situations where it is not appropriate to identify a deletion reason, pass in a value of `null`.
- Even when when a deletion reason is required, you can use a value of `null`. In these cases, the system automatically populates the deletion reason fields with default values. These defaults are: `Other` for deletionReasonCode and `This transaction was deleted by script or web service` for deletionReasonMemo.
- The deletionReason complex type is defined in the core XSD.

For more details about the Use Deletion Reasons feature, see the help topic Recording a Reason for Deleting a Transaction.

Response

The DeleteResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
response	WriteResponse	Contains details on the status of the delete operation and a reference to the deleted record.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Note: If you attempt to delete a record that does not exist, the system returns an INVALID_KEY_OR_REF error.

Sample Code

The following example shows how to delete a cash sale transaction.

C#

```
private void deleteCashSale()
{
   RecordRef myCashSale = new RecordRef();
   myCashSale.internalId = "1745";
   myCashSale.type = RecordType.cashSale;
   myCashSale.typeSpecified = true;

   RecordRef myDeletionReasonCode = new RecordRef();
   myDeletionReasonCode.internalId = "3";

   DeletionReason myDeletionReason = new DeletionReason();
   myDeletionReason.deletionReasonCode = myDeletionReasonCode;
   myDeletionReason.deletionReasonMemo = "Per John in Accounting";

   _service.delete(myCashSale, myDeletionReason);

}
```

SOAP Request

```
<soap:Body>
   <delete xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <baseRef type="cashSale" internalId="1745" xsi:type="q1:RecordRef" xmlns:q1="urn:core_201
7_1.platform.webservices.netsuite.com"/>
         <deletionReason>
            <deletionReasonCode internalId="3" xmlns="urn:core_2017_1.platform.webservices.nets
uite.com"/>
            <deletionReasonMemo xmlns="urn:core_2017_1.platform.webservices.netsuite.com">Per J
ohn in Accounting</deletionReasonMemo>
         </deletionReason>
   </delete>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
   <deleteResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <writeResponse>
         <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.web
services.netsuite.com"/>
         <baseRef xsi:type="platformCore:RecordRef" type="cashSale" internalId="1745" xmlns:pla
tformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </deleteResponse>
</soapenv:Body>
```

deleteList

The deleteList operations is used to delete one or more record instances. The record instances can be of various record types. For example, it is possible to delete a customer and a contact within a single request by using this operation.

Note: An asynchronous equivalent is available for this operation, **asyncDeleteList**. For information about asynchronous request processing, see Synchronous Versus Asynchronous Request Processing.

Request

The DeleteListRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
record[]	RecordRef	An array of recordRef objects that specify the records to be deleted.
deletionReason	DeletionReason	Identifies a deletion reason and deletion reason memo. This parameter supports the Use Deletion Reasons feature, which requires users to provide reasons when they delete transactions. However, even when this feature is not enabled, you must use this parameter with every deleteList request. (You can provide a value of `null` in such cases.) For more details, see Deletion Reason Usage Notes.

Deletion Reason Usage Notes

Note the following about the deletionReason parameter:

- A single deletion reason applies to the entire array of records being deleted. If some of the records are transactions and some are not, then the deletion reason still applies to the transactions. It is ignored for the other records.
- The deletionReason complex type includes two fields: deletionReasonCode and deletionReasonMemo. The deletionReasonCode must identify a deletion reason that is listed at Setup > Accounting > Accounting Lists. If the Use Deletion Reasons feature is enabled and

you use the deletionReasonCode to identify a code that does not exist, the request fails with an `INVALID_REF_KEY` error.

- Deletion reasons can be saved only for transactions. However, in web services, you must use the deletionReason parameter even when referencing other record types, and even when the Use Deletion Reasons feature is not enabled. For situations where it is not appropriate to identify a deletion reason, pass in a value of `null`.
- Even when when a deletion reason is required, you can use a value of `null`. In these cases, the system automatically populates the deletion reason fields with default values. These defaults are: `Other` for deletionReasonCode and `This transaction was deleted by script or web service` for deletionReasonMemo.
- The deletionReason complex type is defined in the core XSD.

For more details about the Use Deletion Reasons feature, see the help topic Recording a Reason for Deleting a Transaction.

Response

The DeleteListResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
response[]	WriteResponse	Contains an array of WriteResponse objects, each of which contains details on the status of the delete operation and a reference to the deleted record.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

The following examples show various ways of using the deleteList operation.

C#

The following example shows how to delete two cash sale transactions. Both transactions have the same deletion reason.

```
private void deleteCashSales()
{
   RecordRef myCashSale = new RecordRef();
   myCashSale.internalId = "1727";
   myCashSale.type = RecordType.cashSale;
   myCashSale.typeSpecified = true;

   RecordRef myCashSale2 = new RecordRef();
   myCashSale2.internalId = "1728";
   myCashSale2.type = RecordType.cashSale;
   myCashSale2.typeSpecified = true;

   RecordRef myDeletionReasonCode = new RecordRef();
   myDeletionReasonCode.internalId = "2";

   DeletionReason myDeletionReason = new DeletionReason();
   myDeletionReason.deletionReasonCode = myDeletionReasonCode;
   myDeletionReason.deletionReasonMemo = "Per Scott";

   RecordRef[] myCashSaleRefs = new RecordRef[2];
   myCashSaleRefs[0] = myCashSale;
   myCashSaleRefs[1] = myCashSale2;

   _service.deleteList(myCashSaleRefs, myDeletionReason);

}
```

The following example shows how to delete two records of different types: an employee record and a calender event record. Because it’s not possible to set a deletion reason for either of those two types of records, the deletion reason used is null.

```
private void deleteVariousRecords()
{
    RecordRef myEmployee = new RecordRef();
    myEmployee.internalId = "360";
    myEmployee.type = RecordType.employee;
    myEmployee.typeSpecified = true;

    RecordRef myCalendarEvent = new RecordRef();
    myCalendarEvent.internalId = "381";
    myCalendarEvent.type = RecordType.calendarEvent;
    myCalendarEvent.typeSpecified = true;

    RecordRef[] myRecords = new RecordRef[2];
    myRecords[0] = myEmployee;
    myRecords[1] = myCalendarEvent;

    _service.deleteList(myRecords, null);

}
```

SOAP Request

The following SOAP request shows how to delete an employee and calendar event record.

```
<soap:Body>
```

```
   <deleteList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <baseRef type="employee" internalId="360" xsi:type="q1:RecordRef" xmlns:q1="urn:core_2017
_1.platform.webservices.netsuite.com"/>
      <baseRef type="calendarEvent" internalId="381" xsi:type="q2:RecordRef" xmlns:q2="urn:core
_2017_1.platform.webservices.netsuite.com"/>
   </deleteList>
</soap:Body>
```

SOAP Response

In the response, the status for each item in the request is returned. If a problem exists with one of the items, creating an error, the other records are still processed.

```
<soapenv:Body>
   <deleteListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <writeResponseList>
         <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.web
services.netsuite.com"/>
            <writeResponse>
               <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
               <baseRef xsi:type="platformCore:RecordRef" type="employee" internalId="360" xmln
s:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </writeResponse>
            <writeResponse>
               <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
               <baseRef xsi:type="platformCore:RecordRef" type="calendarEvent" internalId="381"
 xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </writeResponse>
      </writeResponseList>
   </deleteListResponse>
</soapenv:Body>
```

get

The get operation is used to retrieve a record by providing the unique id that identifies that record.

Request

The getRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
recordRef	RecordRef	A recordRef object that specifies the id of the record to be retrieved.

Response

The getResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
record	Record	A record that represents the specified id. The actual record returned needs to be a type that extends the abstract type Record.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In this example, a single customer record is retrieved. Note that the internal ID for the specific instance of the record and the record type (customer) must be specified.

```
<soap:Body>
<platformMsgs:get>
   <platformMsgs:baseRef internalId="983" type="customer" xsi:type="platformCore:RecordRef">
      <platformCore:name/>
   </platformMsgs:baseRef>
</platformMsgs:get>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
<getResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<readResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"/>

   <record internalId="983" xsi:type="ns2:Customer"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ns2="urn:relationships_2017_1.lists.webservices.netsuite.com">
      <ns2:entityId>Shutter Fly</ns2:entityId><ns2:isInactive>false</ns2:isInactive>
      <ns2:companyName>Shutter Fly, Inc</ns2:companyName>
      <ns2:entityStatus internalId="6">
      .
      ...[more fields]
      .
      <ns2:customFieldList>
         <ns6:customField internalId="custentity_map" xsi:type="ns6:StringCustomFieldRef"
         xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.com">
            <ns6:value>http://maps.google.com</ns6:value>
         </ns6:customField>
      </ns2:customFieldList>
   </record>
</readResponse>
</getResponse>
</soapenv:Body>
```

C#

```
private void getCustomer()
{
   // This operation requires a valid session
   this.login( true );

   // Prompt for the nsKey
   _out.write( "\nnsKey for record to be retrieved: " );
   String nsKey = _out.readLn();

   // Invoke the get() operation to retrieve the record
   RecordRef recordRef = new RecordRef();
   recordRef.internalId = nsKey;
   recordRef.type = RecordType.customer;
   recordRef.typeSpecified = true;

   ReadResponse response = _service.get( recordRef );

   // Process response from get() operation
   _out.info( "\nRecord returned from get() operation: " );
   if ( !response.status.isSuccess )
   {
      _out.info(
      "ERROR: " +
      getStatusDetails( response.status ) );
   }
   else
   {
      Customer customer = (Customer) response.record;
      _out.info(
      "\nnsKey=" + customer.internalId + ", " +
      "\nentityId=" + customer.entityId +
      (customer.companyName==null ? "" : ("\ncompanyName=" + customer.companyName)) +
      (customer.stage==null ? "" : ("\nstage=" + customer.stage)) +
      (customer.email==null ? "" : ("\nemail=" + customer.email)) +
      (customer.phone==null ? "" : ("\nphone=" + customer.phone)) +
      "\nisInactive=" + customer.isInactive +
      (!customer.dateCreatedSpecified ? "" : ("\ndateCreated=" +
      customer.dateCreated.ToShortDateString())) );
   }
}
```

Java

```
public void getCustomer() throws RemoteException, ExceededUsageLimitFault,
UnexpectedErrorFault, InvalidSessionFault, ExceededRecordCountFault {
   // This operation requires a valid session
   this.login(true);

   // Prompt for the nsKey
   _console.write("\nnsKey for record to be retrieved: ");
   String nsKey = _console.readLn();

   // Invoke the get() operation to retrieve the record
   RecordRef recordRef = new RecordRef();
   recordRef.setInternalId(nsKey);
   recordRef.setType(RecordType.customer);

   ReadResponse response = _port.get(recordRef);

   // Process response from get() operation
   _console.info("\nRecord returned from get() operation: ");
   if (!response.getStatus().isIsSuccess()) {
      _console.info("ERROR: " + getStatusDetails(response.getStatus()));
   } else {
      Customer customer = (Customer) response.getRecord();
      _console
      .info("\nnsKey="
      + customer.getInternalId()
      + ", "
      + "\nentityId="
      + customer.getEntityId()
      + (customer.getCompanyName() == null ? ""
      : ("\ncompanyName=" + customer
      .getCompanyName()))
      + (customer.getStage() == null ? ""
      : ("\nstage=" + customer.getStage()))
      + (customer.getEmail() == null ? ""
      : ("\nemail=" + customer.getEmail()))
      + (customer.getPhone() == null ? ""
      : ("\nphone=" + customer.getPhone()))
      + "\nisInactive="
      + customer.getIsInactive()
      + (customer.getDateCreated() != null ? ""
      : ("\ndateCreated=" + customer
      .getDateCreated().toString())));
   }
}
```

getAll

The getAll operation is used to retrieve a list of all records of the specified type. Records that support the getAll operation are listed in the GetAllRecordType, as defined in the platformCoreType system constants XSD file.

Note: You cannot use the search operation to retrieve state values. You must use the getAll operation. The getAll operation will return all states, not the legal ones for your default country. Also note that the country and state must match on the address.

Request

The getAllRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
recordType	GetAllRecordType	Specify the record type.

Response

The getList response type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
recordList	Record[]	A list of records that correspond to the specified ids. The actual records returned need to be of a type that extends the abstract type Record.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

getAsyncResult

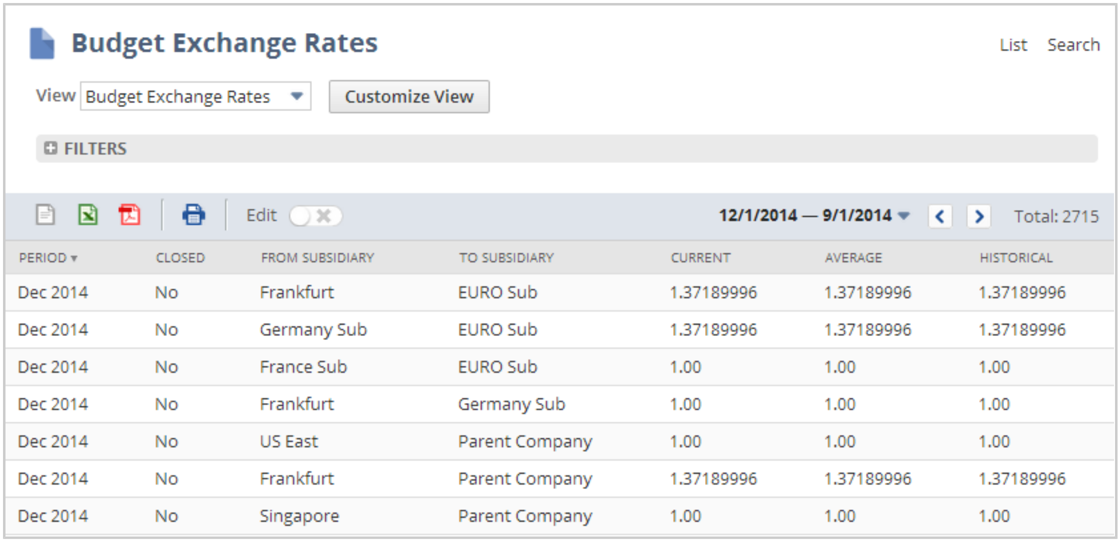
See Checking for Detailed Results of an Asynchronous Job in Asynchronous Request Processing for details.

getBudgetExchangeRate

On the Budget Exchange Rates table, you can maintain exchange rates between the root-parent and child subsidiaries for use in the budgeting process. Use the getBudgetExchangeRate operation to get and filter all data related to this table.

Important: This operation can be used only in NetSuite OneWorld accounts.

In the UI, you can see the Budget Exchange Rates table by going to List > Accounting > Budget Exchange Rates. Note that in web services, this table is read-only.



PERIOD	CLOSED	FROM SUBSIDIARY	TO SUBSIDIARY	CURRENT	AVERAGE	HISTORICAL
Dec 2014	No	Frankfurt	EURO Sub	1.37189996	1.37189996	1.37189996
Dec 2014	No	Germany Sub	EURO Sub	1.37189996	1.37189996	1.37189996
Dec 2014	No	France Sub	EURO Sub	1.00	1.00	1.00
Dec 2014	No	Frankfurt	Germany Sub	1.00	1.00	1.00
Dec 2014	No	US East	Parent Company	1.00	1.00	1.00
Dec 2014	No	Frankfurt	Parent Company	1.37189996	1.37189996	1.37189996
Dec 2014	No	Singapore	Parent Company	1.00	1.00	1.00

For general information on the Budget Exchange Rate table, see the help topic Using Budget Exchange Rates.

Request

The GetBudgetExchangeRateRequest type is used for the request.

Element Name	XSD Type	Notes
budgetExchangeRateFilter	BudgetExchageRateFilter	You can filter the returned exchange rates for a budget using this filter.

BudgetExchangeRateFilter

Element Name	XSD Type	Notes
period	RecordRef	References an existing period. This argument is required.
fromSubsidiary	RecordRef	References the receiving subsidiary. This argument is optional.

Element Name	XSD Type	Notes
toSubsidiary	RecordRef	References the originating subsidiary. This argument is optional.

Response

The GetBudgetExchangeRateResult type is used for the response.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
budgetExchangeRateList	BudgetExchangeRateList	Returns a list of available exchange rates in a budget.

BudgetExchangeRateList

Element Name	XSD Type	Notes
budgetExchangeRate	BudgetExchangeRate	References the exchange rate for a budget.

BudgetExchangeRate

Element Name	XSD Type	Notes
period	RecordRef	References an existing period.
fromSubsidiary	RecordRef	References the receiving subsidiary.
toSubsidiary	RecordRef	References the originating subsidiary.
currentRate	double	References the current rate.
averageRate	double	References the average rate.
historicalRate	double	References the historical rate.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http:
//www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <soapenv:Body>
         <getBudgetExchangeRate xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">

            <budgetExchangeRateFilter>
               <ns1:period internalId="3" xmlns:ns1="urn:core_2017_1.platform.webservices.netsu
ite.com"/>
               <ns2:fromSubsidiary internalId="4" xmlns:ns2="urn:core_2017_1.platform.webservic
es.
      netsuite.com"/>
            </budgetExchangeRateFilter>
         </getBudgetExchangeRate>
      </soapenv:Body>
   </soapenv:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="utf-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http:
//www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <soapenv:Body>
         <getBudgetExchangeRateResponse xmlns="urn:messages_2017_1.platform.webservices.netsuit
e.com">
            <platformCore:getBudgetExchangeRateResult
      xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
               <platformCore:status isSuccess="true"/>
               <platformCore:budgetExchangeRateList>
                  <platformCore:budgetExchangeRate>
                     <platformCore:period internalId="3"/>
                     <platformCore:fromSubsidiary internalId="4"/>
                     <platformCore:toSubsidiary internalId="1"/>
                     <platformCore:currentRate>1.9586</platformCore:currentRate>
                     <platformCore:averageRate>1.9586</platformCore:averageRate>
                     <platformCore:historicalRate>1.9586</platformCore:historicalRate>
                  </platformCore:budgetExchangeRate>
               </platformCore:budgetExchangeRateList>
            </platformCore:getBudgetExchangeRateResult>
         </getBudgetExchangeRateResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Java

```
/* Make Record Ref out of an internalId */
   public static RecordRef mrr(String internalId)
```

```
    {
       RecordRef toRet = new RecordRef();
       toRet.setInternalId(internalId);
       return toRet;
    }

c.getPort().getBudgetExchangeRate(new BudgetExchangeRateFilter(mrr("3"),mrr("4"),null));
```

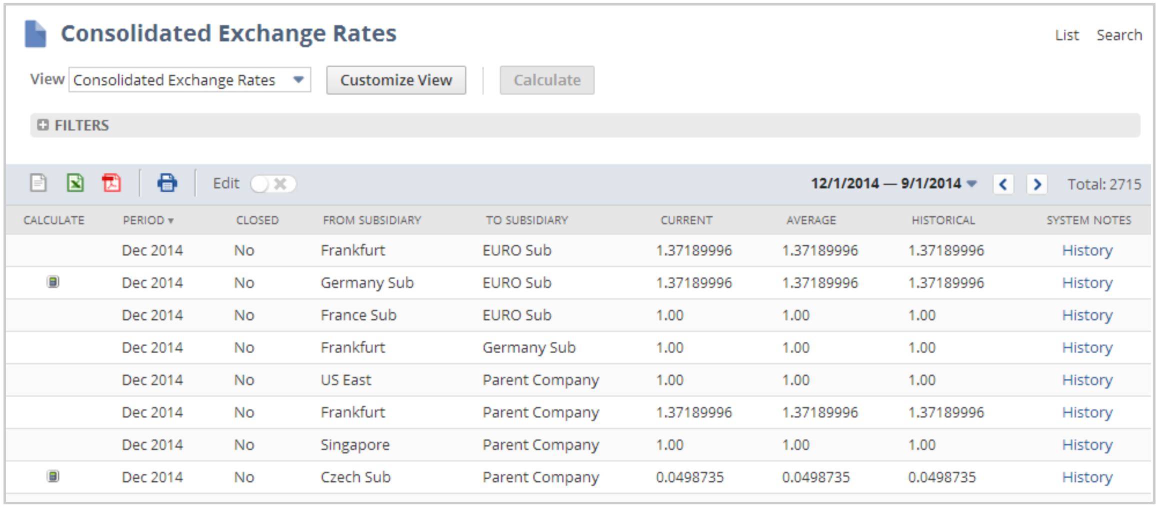
getConsolidatedExchangeRate

On the Consolidated Exchange Rates table, you can maintain exchange rates between the root-parent and child subsidiaries. Use the getConsolidatedExchangeRate operation to get and filter all data related to this table.

Important: The exposure of the consolidated exchange rate record prompted the removal of the getConsolidatedExchangeRate operation from the 2017.1 endpoint. You can still use the getConsolidatedExchangeRate operation on earlier endpoints. For more information about working with consolidated exchange rates on endpoints starting from 2017.1, see the help topic Consolidated Exchange Rate.

Important: This operation can be used only in NetSuite OneWorld acccounts.

In the UI, you can see the Consolidated Exchange Rates table by going to List > Accounting > Consolidated Exchange Rates. Note that in web services, this table is read-only.



CALCULATE	PERIOD	CLOSED	FROM SUBSIDIARY	TO SUBSIDIARY	CURRENT	AVERAGE	HISTORICAL	SYSTEM NOTES
	Dec 2014	No	Frankfurt	EURO Sub	1.37189996	1.37189996	1.37189996	History
■	Dec 2014	No	Germany Sub	EURO Sub	1.37189996	1.37189996	1.37189996	History
	Dec 2014	No	France Sub	EURO Sub	1.00	1.00	1.00	History
	Dec 2014	No	Frankfurt	Germany Sub	1.00	1.00	1.00	History
	Dec 2014	No	US East	Parent Company	1.00	1.00	1.00	History
	Dec 2014	No	Frankfurt	Parent Company	1.37189996	1.37189996	1.37189996	History
	Dec 2014	No	Singapore	Parent Company	1.00	1.00	1.00	History
■	Dec 2014	No	Czech Sub	Parent Company	0.0498735	0.0498735	0.0498735	History

For general information on currencies and working with the Consolidated Exchange Rate table, see the following topics:

- Creating Currency Records
- Setting Currency Exchange Rates

Note: If you choose, you can set exchange rates in the Consolidated Exchange Rates table directly. See the steps under "To set exchange rates in the Consolidated Exchange Rates table" for details.

- Using the Currency Exchange Rate Integration Feature

- Working with Currencies

getConsolidatedExchangeRate for the Multi-Book Accounting Feature

Results for the getConsolidatedExchageRate operation can include values for the book field. Results can also be filtered by this field.

- If the Multi-Book Accounting feature is enabled:
  - If the book is not set in the request, getConsolidatedExchangeRate returns values for all accounting books that have the Enable Consolidation option selected.
  - If the book is set in the request, getConsolidatedExchangeRate returns values for the specified accounting book.
- If the Multi-Book Accounting feature is not enabled:
  - If the book is set in the request, getConsolidatedExchangeRate returns an error with the code = "FEATURE_DISABLED" and the message = "The ''Multi-Book Accounting'' feature is not enabled in your NetSuite account."
  - Book is not part of the response.

Note: In endpoints prior to 2016.2, the getConsolidatedExchangeRate operation can only be used to return results for the primary accounting book, even if the Multi-Book Accounting feature is enabled.

Request

The GetConsolidatedExchangeRateRequest type is used for the request.

Element Name	XSD Type	Notes
consolidatedExchangeRateFilter	ConsolidatedExchangeRateFilter	You can filter the returned consolidated exchange rates using this filter.

ConsolidatedExchangeRateFilter

ConsolidatedExchangeRateFilter is defined in the core XSD.

Element Name	XSD Type	Notes
period	RecordRef	References an existing period. This argument is required.
fromSubsidiary	RecordRef	References the receiving subsidiary. This argument is optional.
toSubsidiary	RecordRef	References the originating subsidiary. This argument is optional.
book	RecordRef	References the specific accounting book for accounts with the Multi-Book Accounting feature enabled.

Response

The GetConsolidatedExchangeRateResult type is used for the response.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
consolidatedExchangeRateList	ConsolidatedExchangeRateList	Returns a list of available consolidated exchange rates.

ConsolidatedExchangeRateList

Element Name	XSD Type	Notes
consolidatedExchangeRate	ConsolidatedExchangeRate	References an existing period.

ConsolidatedExchangeRate

Element Name	XSD Type	Notes
period	RecordRef	References an existing period.
fromSubsidiary	RecordRef	References the receiving subsidiary.
toSubsidiary	RecordRef	References the originating subsidiary.
currentRate	double	References the current rate.
averageRate	double	References the average rate.
historicalRate	double	References the historical rate.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

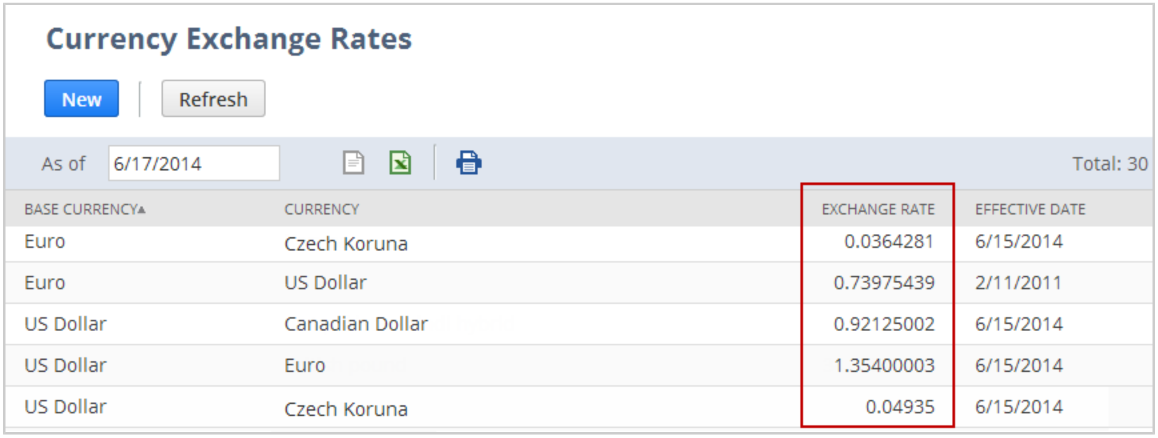
The getConsolidatedExchangeRate and getBudgetExchangeRate operations are used in the same way. Therefore, please refer to the Sample Code for getBudgetExchangeRate.

getCurrencyRate

Use this operation to get the exchange rate between two currencies based on a certain date. The exchange rate values you are getting are those that appear in the Exchange Rate column of the Currency Exchange Rates table, which you can view at Lists > Accounting > Currency Exchange Rates.

The role used to execute this operation must have at least View level of the Currency permission (a Lists type permission).

Note: The Multiple Currencies feature must be enabled in your account before using the getCurrencyRate operation. For information on enabling this feature, see the help topic Enabling the Multiple Currencies Feature.



BASE CURRENCY▲	CURRENCY	EXCHANGE RATE	EFFECTIVE DATE
Euro	Czech Koruna	0.0364281	6/15/2014
Euro	US Dollar	0.73975439	2/11/2011
US Dollar	Canadian Dollar	0.92125002	6/15/2014
US Dollar	Euro	1.35400003	6/15/2014
US Dollar	Czech Koruna	0.04935	6/15/2014

Request

The GetCurrencyRateRequest type is used for the request. This method accepts the argument CurrencyRateFilter. With this filter you specify what you want returned in the results.

If you do not specify anything in the filter, the resulting CurrencyRateList will contain all currency exchange rates (all combinations) for the actual date. For example, if you specify baseCurrency only (for example, Euro), you will get all rates for Euro {USD,YEN,SING,....}.

Element Name	XSD Type	Notes
currencyRateFilter	CurrencyRateFilter	Filter the returned currency exchange rates using this filter.

CurrencyRateFilter

Element Name	XSD Type	Notes
baseCurrency	RecordRef	References the base currency. This argument is optional.
fromCurrency	RecordRef	References the from or “exchange” currency. This argument is optional.
effectiveDate	dateTime	This argument is optional. If a date is not provided, the date defaults to the current date.

Response

The GetCurrencyRateResponse type is used for the response.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
currencyRate	double	Returns a list of available currency exchange rates.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- InsufficientPermissionFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request

```
<soapenv:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/200
1/XMLSchema-instance" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Header>
        <ns1:passport soapenv:mustUnderstand="0" soapenv:actor="http://schemas.xmlsoap.org/soap
/actor/next" xmlns:ns1="urn:messages_2017_1.platform.webservices.netsuite.com">
            <ns2:email xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">email@nets
uite.com</ns2:email>
            <ns3:password xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">*******
</ns3:password>
            <ns4:account xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com">1234567<
/ns4:account>
            <ns5:role internalId="3" xmlns:ns5="urn:core_2017_1.platform.webservices.netsuite.c
om"/>
        </ns1:passport>
    </soapenv:Header>
    <soapenv:Body>
        <getCurrencyRate xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <currencyRateFilter>
                <ns6:baseCurrency internalId="2" xmlns:ns6="urn:core_2017_1.platform.webservice
s.netsuite.com"/>
                <ns7:fromCurrency internalId="1" xmlns:ns7="urn:core_2017_1.platform.webservice
```

```
s.netsuite.com">
                <ns8:effectiveDate xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com
">2012-08
         11T06:58:59.628Z</ns8:effectiveDate>
            </currencyRateFilter>
        </getCurrencyRate>
    </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```
<soapenv:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/200
1/XMLSchema-instance" xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Header>
        <platformMsgs:documentInfo xmlns:platformMsgs="urn:messages_2017_1.platform.webservices
.netsuite.com">
            <platformMsgs:nsId>WEBSERVICES_1326288_080920122571729847762165376c7c330232bb5</pl
atformMsgs:nsId>
        </platformMsgs:documentInfo>
    </soapenv:Header>
    <soapenv:Body>
        <getCurrencyRateResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">

            <platformCore:getCurrencyRateResult xmlns:platformCore="urn:core_2017_1.platform.we
bservices.netsuite.com">
                <platformCore:status isSuccess="true"/>
                <platformCore:currencyRateList>
                    <platformCore:currencyRate>
                        <platformCore:baseCurrency internalId="2"/>
                        <platformCore:fromCurrency internalId="1"/>
                        <platformCore:exchangeRate>0.42283298</platformCore:exchangeRate>
                        <platformCore:effectiveDate>1970-01-01T00:00:00.000-08:00</platformCore
:effectiveDate>
                    </platformCore:currencyRate>
                </platformCore:currencyRateList>
            </platformCore:getCurrencyRateResult>
        </getCurrencyRateResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Java

This sample shows how to get the actual rate for tomorrow and for USD to GBP.

```
    public void getCurrencyRate() throws Exception
   {
      this.login();

      CurrencyRateFilter crf = new CurrencyRateFilter();
      Calendar cal = Calendar.getInstance();
      cal.add(Calendar.DAY_OF_YEAR, +1); //tomorrow
      crf.setEffectiveDate(cal);
```

```
        RecordRef currencyRef1 = new RecordRef();
        currencyRef1.setInternalId("1"); //USD
        crf.setBaseCurrency(currencyRef1);

        RecordRef currencyRef2 = new RecordRef();
        currencyRef2.setInternalId("2"); //GBP
        crf.setFromCurrency(currencyRef2);

        GetCurrencyRateResult gcrr = _port.getCurrencyRate(crf);
        CurrencyRateList crl = gcrr.getCurrencyRateList();
        CurrencyRate cr = crl.getCurrencyRate(0);
        Double rate = cr.getExchangeRate();

        System.out.println("Exchange rate is = " + rate);
    }
```

Note: The mrr() method is a convenience method for specifying a record without the need to indicate a record type.

```
/* Make Record Ref out of an internalId */
        public static RecordRef mrr(String internalId)
        {
                RecordRef toRet = new RecordRef();
                toRet.setInternalId(internalId);
                return toRet;
        }
```

getCustomizationId

When integrating with NetSuite through web services, in many cases, you will want to know which custom objects exist in an account. You may also want to obtain metadata for these custom objects so that your application can handle any business logic that is specific to the account.

To learn which custom objects exist in an account, use the getCustomizationId operation to retrieve the internalIds, externalIds, and/or scriptIds of all custom objects of a specified type. These types, enumerated in CustomizationType, include the following:

- crmCustomField
- customList
- customRecordType
- entityCustomField
- itemCustomField
- itemOptionCustomField
- otherCustomField
- transactionBodyCustomField
- transactionColumnCustomField

Note: The CustomizationType object is defined in the coreTypes XSD.

After the IDs are returned, you can then pass the ID value(s) to either the get or getList operation to obtain metadata for specific custom objects (see Using get and getList with getCustomizationId for more details). If you are returning large sets of metadata, the getCustomizationId operation lets you add pagination to break your response into multiple pieces.

Note the following:

- To see the UI equivalent of a NetSuite custom object, go to Customization > Lists, Records, & Fields > [custom object type]. For information on working with each object type in web services, see the help topic SuiteBuilder Overview. For a more general understanding of the NetSuite customization model, see the help topic SuiteBuilder (Customization).
- Normally, you cannot add or update the internalId of a NetSuite object. Custom objects, however, are an exception to this rule. You can specify the ID on add, but you cannot update the ID thereafter. Also note that internalIds for custom objects can be set to any unique alphanumeric string up to 30 characters long. This string cannot include any spaces, but it can include underscores (_).

Using get and getList with getCustomizationId

If you want only the internalIds, externalIds, and/or scriptIds for custom objects of a specific type, you can call getCustomizationId on that custom type (for example, crmCustomField). However, if you also want the metadata for any or all of the custom objects referenced in the getCustomizationId response, you must invoke a get or getList request and specify each object's ID.

When using either the get or getList operations in conjunction with getCustomizationId, be aware of the following:

- If you specify more than one ID in a get | getList call, the IDs you pass are read in the following order: internalId, externalId, scriptId.
- If an invalid scriptId is specified but the internalId or externalId is correct, the scriptId is ignored.
- NetSuite will reject get | getList requests if only a scriptId is passed for record types that do not support scriptIds.

Request

The getCustomizationIdRequest type is used for this request. It contains the following fields:

Element Name	XSD Type	Notes
customizationType	CustomizationType	Any of the custom object types enumerated in CustomizationType.
includeInactives	boolean	A value of true or false is required. A value of false means no inactive custom objects are returned in the response. A value of true means that both active and inactive custom objects are returned in the response. ■ To determine which of the returned custom objects are inactive, you must perform either a get or getList operation to obtain the metadata for the custom object type(s). For custom records and custom lists, the value is specified in the **isInactive** element. ■ Although the includeInactives argument is required, the value you set applies to custom record and custom list objects only.

Element Name	XSD Type	Notes
		In NetSuite, custom fields cannot be set to active or inactive. Therefore, when using getCustomizationId on a custom field type, the value you specify for includeInactives will be ignored. Whether you specify true or false, all custom field objects (of the type specified in your request) will be returned.

Response

The getCustomizationIdResult type is used for the response. It contains the following fields:

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
totalRecords	xsd:int	The total number of records for this request. Depending on the pageSize value, some or all the records may be returned in the response.
customizationRefList	CustomizationRef	A list of custom objects that correspond to the specified customization type. Also returns the internalId, externalId, and/or scriptId of each object.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

The following sample is a two-part sample. The first part of the sample shows how to construct a getCustomizationId request so that the IDs for all custom record type objects in a NetSuite account are returned. Notice that in the getCustomizationId request, the value of the includeInactives parameter is set to false, meaning that only custom records marked as “active” in the account will be returned.

The second part of the sample shows how to take internalIds that are returned, and make a getList request to return the metadata for each custom record type.

C# Sample

```
NetSuiteService nss = new NetSuiteService();

// credential code ignored here in this sample ...

// Perform getCustomizationId on custom record type
CustomizationType ct = new CustomizationType();
ct.getCustomizationTypeSpecified=true;
ct.getCustomizationType = GetCustomizationType.customRecordType;

// Retrieve active custom record type IDs. The includeInactives param is set to false.
GetCustomizationIdResult getCustIdResult = nss.getCustomizationId(ct, false);

// Retrieve the metadata of the returned custom record types
ReadResponse [] readResp = nss.getList(getCustIdResult.customizationRefList);
```

SOAP Request (getCustomizationId)

```
         <getCustomizationId xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <customizationType getCustomizationType="customRecordType" />
            <includeInactives>false</includeInactives>
         </getCustomizationId>
```

SOAP Request (getList)

```
            <getList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
               <baseRef xmlns:q1="urn:core_2017_1.platform.webservices.netsuite.com"
      xsi:type="q1:CustomizationRef" internalId="15" scriptId="customrecord15"
         type="customRecordType">
                  <q1:name>Customer Satisfaction Survey</q1:name>
               </baseRef>
            </getList>
```

SOAP Response (getCustomizationId)

```
<getCustomizationIdResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <platformCore:getCustomizationIdResult
      xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
               <platformCore:status isSuccess="true"/>
               <platformCore:totalRecords>1</platformCore:totalRecords>
               <platformCore:customizationRefList>
                  <platformCore:customizationRef internalId="15" scriptId="customrecord15"
         type="customRecordType">
                     <platformCore:name>Customer Satisfaction Survey</platformCore:name>
                  </platformCore:customizationRef>
               </platformCore:customizationRefList>
            </platformCore:getCustomizationIdResult>
         </getCustomizationIdResponse>
```

SOAP Response (getList)

```
 <getListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
              <readResponseList>
                 <readResponse>
                    <platformCore:status isSuccess="true"
        xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
                    <record internalId="15" xsi:type="setupCustom:CustomRecordType"
        xmlns:setupCustom="urn:customization_2017_1.setup.webservices.netsuite.com">
                       <setupCustom:recordName>Customer Satisfaction Survey</setupCustom:recor
dName>
                       <setupCustom:includeName>true</setupCustom:includeName>
                       <setupCustom:showId>true</setupCustom:showId>
                       <setupCustom:showCreationDate>true</setupCustom:showCreationDate>
                       <setupCustom:showCreationDateOnList>true</setupCustom:showCreationDateO
nList>
                       <setupCustom:showLastModified>true</setupCustom:showLastModified>
                       <setupCustom:showLastModifiedOnList>true</setupCustom:showLastModifiedO
nList>
                       <setupCustom:showOwner>false</setupCustom:showOwner>
                       <setupCustom:showOwnerOnList>false</setupCustom:showOwnerOnList>
                       <setupCustom:showOwnerAllowChange>false</setupCustom:showOwnerAllowChan
ge>
                       <setupCustom:usePermissions>false</setupCustom:usePermissions>
                       <setupCustom:allowAttachments>false</setupCustom:allowAttachments>
                       <setupCustom:showNotes>true</setupCustom:showNotes>
                       <setupCustom:enableMailMerge>false</setupCustom:enableMailMerge>
                       <setupCustom:isOrdered>false</setupCustom:isOrdered>
                       <setupCustom:allowInlineEditing>false</setupCustom:allowInlineEditing>
                       <setupCustom:isAvailableOffline>false</setupCustom:isAvailableOffline>
                       <setupCustom:allowQuickSearch>false</setupCustom:allowQuickSearch>
                       <setupCustom:isInactive>false</setupCustom:isInactive>
                       <setupCustom:disclaimer><b>Record numbers cannot be reverted back to na
mes after being
            updated.<br/></setupCustom:disclaimer>
                       <setupCustom:enableNumbering>false</setupCustom:enableNumbering>
                       <setupCustom:numberingCurrentNumber>413</setupCustom:numberingCurrentNu
mber>
                       <setupCustom:allowNumberingOverride>false</setupCustom:allowNumberingOv
erride>
                       <setupCustom:isNumberingUpdateable>false</setupCustom:isNumberingUpdate
able>
                       <setupCustom:scriptId>customrecord15</setupCustom:scriptId>
                       <setupCustom:fieldList>
                          <setupCustom:customField scriptId="CUSTRECORD_CONTACT_TYPE">
                             <setupCustom:label>Contact type</setupCustom:label>
                             <setupCustom:fieldType>_listRecord</setupCustom:fieldType>
                             <setupCustom:selectRecordType internalId="16"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                <platformCore:name>Customer contact type</platformCore:name>
                             </setupCustom:selectRecordType>
                             <setupCustom:storeValue>true</setupCustom:storeValue>
                             <setupCustom:showInList>false</setupCustom:showInList>
                             <setupCustom:globalSearch>false</setupCustom:globalSearch>
                             <setupCustom:isParent>false</setupCustom:isParent>
```

```
                  <setupCustom:displayType>_normal</setupCustom:displayType>
                  <setupCustom:isMandatory>true</setupCustom:isMandatory>
                  <setupCustom:checkSpelling>false</setupCustom:checkSpelling>
                  <setupCustom:defaultChecked>false</setupCustom:defaultChecked>
                  <setupCustom:isFormula>false</setupCustom:isFormula>
                  <setupCustom:recType>15</setupCustom:recType>
                  <setupCustom:roleRestrict>false</setupCustom:roleRestrict>
                  <setupCustom:accessLevel>_edit</setupCustom:accessLevel>
                  <setupCustom:searchLevel>_edit</setupCustom:searchLevel>
               </setupCustom:customField>
               <setupCustom:customField scriptId="CUSTRECORD_IF_OTHER">
                  <setupCustom:label>Other contact type</setupCustom:label>
                  <setupCustom:fieldType>_freeFormText</setupCustom:fieldType>
                  <setupCustom:storeValue>true</setupCustom:storeValue>
                  <setupCustom:showInList>false</setupCustom:showInList>
                  <setupCustom:globalSearch>false</setupCustom:globalSearch>
                  <setupCustom:isParent>false</setupCustom:isParent>
                  <setupCustom:displayType>_normal</setupCustom:displayType>
                  <setupCustom:displayWidth>42</setupCustom:displayWidth>
                  <setupCustom:isMandatory>false</setupCustom:isMandatory>
                  <setupCustom:checkSpelling>false</setupCustom:checkSpelling>
                  <setupCustom:defaultChecked>false</setupCustom:defaultChecked>
                  <setupCustom:isFormula>false</setupCustom:isFormula>
                  <setupCustom:recType>15</setupCustom:recType>
                  <setupCustom:roleRestrict>false</setupCustom:roleRestrict>
                  <setupCustom:accessLevel>_edit</setupCustom:accessLevel>
                  <setupCustom:searchLevel>_edit</setupCustom:searchLevel>
               </setupCustom:customField>

// custom field list continues ...

         </setupCustom:fieldList>
         <setupCustom:formsList>
               <setupCustom:forms>
                  <setupCustom:formName>Standard Customer Satisfaction Survey
Form<setupCustom:formName>
                  <setupCustom:formPref>false</setupCustom:formPref>
               </setupCustom:forms>
            </setupCustom:formsList>
            <setupCustom:parentsList>
               <setupCustom:parents>
                  <setupCustom:childDescr>Customer</setupCustom:childDescr>
               </setupCustom:parents>
            </setupCustom:parentsList>
         </record>
      </readResponse>
   </readResponseList>
</getListResponse>
```

getDataCenterUrls

You use the getDataCenterUrls operation to obtain the correct URL for external client access to NetSuite.

This operation is critical because NetSuite hosts customer accounts in multiple data centers. The correct URL for web services access varies depending on the data center hosting the account. Your integration **must** incorporate logic that dynamically determines the correct URL. With the 2012.2 and later endpoints, you should use the getDataCenterUrls operation to dynamically discover the correct URL. With older endpoints, you should use the REST roles service or, starting from version 2017.2, the DataCenterUrls REST service.

Important: If you receive an error, try using the c parameter in your request, and specify your account ID. The c (version) parameter ensures that your request is routed correctly according to your version and that you receive the correct response.

In addition to the URL for web services access, the getDataCentersUrls operation also supports discovery of other NetSuite domains. You can use this operation to discover any of the following:

- restDomain – https://rest.netsuite.com, https://rest.na1.netsuite.com, and similar domains
- systemDomain – https://system.netsuite.com, https://system.na1.netsuite.com, and similar domains
- webservicesDomain – https://webservices.netsuite.com, https://webservices.na1.netsuite.com, and similar domains

If you are building an integration for your own use or for a single customer, you integration must include logic to determine the correct URL for the appropriate NetSuite account. If you are building an application that will be distributed to multiple customers, you must incorporate logic that can determine the correct URL for any of those customers.

Be aware that getDataCenterURLs requests do not require authentication, nor any information identifying the application.

Important: Your web services integration **must** incorporate logic that dynamically determines the correct URL for web services access.

Note: As of 2017.2, account-specific domains are supported for web services, and you can access your web services domain at the following URL, where 123456 is your account ID: 123456.suitetalk.api.netsuite.com. The data center-specific domains supported before 2017.2 will continue to be supported. For more information, see the help topic URLs for Account-Specific Domains.

REST Services as Alternate Methods for Dynamic Discovery of URL

The getDataCenterUrls operation is only available in the 2012.2 and later endpoints. However you do not have to upgrade your endpoint to support dynamic discovery of the correct web services domain. NetSuite also provides REST services that you can use.

- For general information about the DataCenterUrls REST service, see the help topic DataCenterUrls REST Service.
- For general information about the REST roles service, see the help topic Using the REST roles Service to Get User Accounts, Roles, and Domains.
- For example code that uses this REST service, see:
  - Java REST Example for Web Services Domain Discovery
  - .NET REST Example for Web Services Domain Discovery

  Note that NetSuite Developer Resources sample applications also incorporate RESTful code for dynamic discovery of web services domain. These samples are available at the SuiteTalk Sample Applications Page.

Web Services URLs for Sandbox, EU Sandbox, and Release Preview Environments

For information on the correct URL to use for web services access to sandbox, EU sandbox, and Release Preview environments, see the help topic Understanding NetSuite URLs and Data Centers.

Request

The GetDataCenterUrlsRequest type is used for this request. It requires the following field:

Element Name	XSD Type	Notes
account	string	NetSuite account ID. **Note:** If an invalid account ID is specified, the default URLs are returned.

Authenticated requests contain the following fields in addition to account:

Element Name	XSD Type	Notes
email	string	Email used to log in to NetSuite.
password	string	Password used to log in to NetSuite.
role	RecordRef	References an existing role for the user.

Response

The GetDataCenterUrlsResult type is used for the response. It contains the following fields:

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
restDomain	string	URL used by external client applications to access NetSuite through a RESTlet.
systemDomain	string	
webservicesDomain	string	URL used by external client applications to access NetSuite through web services.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- ExceededRequestSizeFault
- UnexpectedErrorFault

Also, a DATA_REQD error is thrown if no account ID is provided in the request. For an example, see SOAP Response (empty account).

Sample Code

C# Sample

```
    class DataCenterAwareNetSuiteService : NetSuiteService
    {
        public DataCenterAwareNetSuiteService(string account)
            : base()
        {
            System.Uri originalUri = new System.Uri(this.Url);
         DataCenterUrls urls = getDataCenterUrls(account).dataCenterUrls;
            Uri dataCenterUri = new Uri(urls.webservicesDomain + originalUri.PathAndQuery);
            this.Url = dataCenterUri.ToString();
        }
    }
```

Java Sample (public)

```
/**
 * Since 12.2 endpoint accounts are located in multiple data centers with different domain name
s.
 * To use the correct one, wrap the Locator and get the correct domain first.
 *
 * See getDataCenterUrls WSDL method documentation in the Help Center.
 */
private static class DataCenterAwareNetSuiteServiceLocator extends NetSuiteServiceLocator
{
   private String account;

   public DataCenterAwareNetSuiteServiceLocator(String account)
   {
      this.account = account;
   }

   @Override
   public NetSuitePortType getNetSuitePort(URL defaultWsDomainURL)
   {
      try
      {
         NetSuitePortType _port = super.getNetSuitePort(defaultWsDomainURL);
         // Get the webservices domain for your account
         DataCenterUrls urls = _port.getDataCenterUrls(account).getDataCenterUrls();
         String wsDomain = urls.getWebservicesDomain();

         // Return URL appropriate for the specific account
         return super.getNetSuitePort(new URL(wsDomain.concat(defaultWsDomainURL.getPath())));
      }
      catch (Exception e)
      {
         throw new RuntimeException(e);
      }
   }
}
```

```
---
// Locate the NetSuite web service.
NetSuiteServiceLocator service = new DataCenterAwareNetSuiteServiceLocator("myAccount");
// Get the service port (to the correct data center)
_port = service.getNetSuitePort(new URL("https://webservices.netsuite.com/services/NetSuitePort
_2012_2"));
```

Java Sample (authenticated)

```
public void getDataCenterUrlsAuthenticated() throws Exception
   {
      this.login();

      String account = "1326288";
      System.out.println("Account: " + account);
      DataCenterUrls urls = _port.getDataCenterUrls(account).getDataCenterUrls();
      System.out.println("\tREST domain: " + urls.getRestDomain());
      System.out.println("\tWeb services domain: " + urls.getWebservicesDomain());
      System.out.println("\tSystem domain: " + urls.getSystemDomain());
      System.out.println();
   }
```

Java Sample (public)

```
public void testDataCenterUrlsPublic() throws Exception
{
    NLWsClient c = new NLWsClient();
    c.initPort();
    String account = "1326288";


    System.out.println("Account: " + account);
    DataCenterUrls urls = c.getPort().getDataCenterUrls(account).getDataCenterUrls();


    System.out.println("\tREST domain: " + urls.getRestDomain());
    System.out.println("\tWeb services domain: " + urls.getWebservicesDomain());
    System.out.println("\tSystem domain: " + urls.getSystemDomain());
    System.out.println();
}
```

Java Sample (authenticated)

```
public void testDataCenterUrlsAuthenticated() throws Exception
{
    NLWsClient c = new NLWsClient(new String[]{"happygilmore@abc.com","password1","1326288","3"
},443,"https://webservices.netsuite.com"});
    c.useRequestLevelCredentials();
    String account = "1326288";
    System.out.println("Account: " + account);
    DataCenterUrls urls = c.getPort().getDataCenterUrls(account).getDataCenterUrls();


    System.out.println("\tREST domain: " + urls.getRestDomain());
    System.out.println("\tweb services domain: " + urls.getWebservicesDomain());
    System.out.println("\tSystem domain: " + urls.getSystemDomain());
    System.out.println();
}
```

SOAP Request (public)

```
<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http:
//www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <soapenv:Body>
         <getDataCenterUrls xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <account>1326288</account>
         </getDataCenterUrls>
      </soapenv:Body>
   </soapenv:Envelope>
```

SOAP Response (public)

```
<?xml version="1.0" encoding="utf-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http:
//www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <soapenv:Header>
         <platformMsgs:documentInfo xmlns:platformMsgs="urn:messages_2017_1.platform.webservice
s.netsuite.com">
            <platformMsgs:nsId>WEBSERVICES_1326288_081620127905218732602645554_10fa27f165429</pl
atformMsgs:nsId>
         </platformMsgs:documentInfo>
      </soapenv:Header>
      <soapenv:Body>
         <getDataCenterUrlsResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.co
m">
            <getDataCenterUrlsReturn xsi:type="GetDataCenterUrlsResult" xmlns:platformCore="urn
:core_2017_1.platform.webservices.netsuite.com">
               <platformCore:status isSuccess="true"/>
               <platformCore:dataCenterUrls>
                  <platformCore:restDomain>https://<accountID>.restlets.api.netsuite.com</platf
ormCore:restDomain>
                  <platformCore:webservicesDomain>https://<accountID>.suitetalk.api.netsuite.co
m</platformCore:webservicesDomain>
                  <platformCore:systemDomain>https://system.netsuite.com</platformCore:systemDo
main>
               </platformCore:dataCenterUrls>
            </getDataCenterUrlsReturn>
         </getDataCenterUrlsResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

SOAP Request (authenticated)

```
<?xml version="1.0" encoding="UTF-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http:
//www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <soapenv:Header>
         <ns1:passport soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next" soapenv:mustU
nderstand="0" xmlns:ns1="urn:messages_2017_1.platform.webservices.netsuite.com">
            <ns2:email xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">happygilmo
re@abc.com</ns2:email>
            <ns3:password xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">passwor
d1</ns3:password>
            <ns4:account xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com">1326288<
/ns4:account>
            <ns5:role internalId="3" xmlns:ns5="urn:core_2017_1.platform.webservices.netsuite.c
om"/>
         </ns1:passport>
      </soapenv:Header>
      <soapenv:Body>
         <getDataCenterUrls xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <account>1326288</account>
         </getDataCenterUrls>
      </soapenv:Body>
   </soapenv:Envelope>
```

SOAP Response (authenticated)

```
<?xml version="1.0" encoding="utf-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http:
//www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <soapenv:Header>
         <platformMsgs:documentInfo xmlns:platformMsgs="urn:messages_2017_1.platform.webservice
s.netsuite.com">
            <platformMsgs:nsId>WEBSERVICES_1326288_0816201279052187326026 4554_10fa27f165429</pl
atformMsgs:nsId>
         </platformMsgs:documentInfo>
      </soapenv:Header>
      <soapenv:Body>
         <getDataCenterUrlsResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.co
m">
            <getDataCenterUrlsReturn xsi:type="GetDataCenterUrlsResult" xmlns:platformCore="urn
:core_2017_1.platform.webservices.netsuite.com">
               <platformCore:status isSuccess="true"/>
               <platformCore:dataCenterUrls>
                  <platformCore:restDomain>https://<accountID>.restlets.api.netsuite.com</platf
ormCore:restDomain>
                  <platformCore:webservicesDomain>https://<accountID>.suitetalk.api.netsuite.co
m</platformCore:webservicesDomain>
                  <platformCore:systemDomain>https://system.netsuite.com</platformCore:systemDo
main>
               </platformCore:dataCenterUrls>
            </getDataCenterUrlsReturn>
         </getDataCenterUrlsResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

SOAP Response (empty account)

```
<?xml version="1.0" encoding="utf-8"?>
   <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http:
//www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <soapenv:Header>
         <platformMsgs:documentInfo xmlns:platformMsgs="urn:messages_2017_1.platform.webservice
s.netsuite.com">
            <platformMsgs:nsId>WEBSERVICES__0911201212589142971204370270_d03cd34b349479541b</pl
atformMsgs:nsId>
         </platformMsgs:documentInfo>
      </soapenv:Header>
      <soapenv:Body>
         <getDataCenterUrlsResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.co
m">
            <getDataCenterUrlsReturn xsi:type="GetDataCenterUrlsResult" xmlns:platformCore="urn
:core_2017_1.platform.webservices.netsuite.com">
               <platformCore:status isSuccess="false">
                  <platformCore:statusDetail type="ERROR">
                     <platformCore:code>DATA_REQD</platformCore:code>
                     <platformCore:message>You need to provide a proper value for the required
field: account.</platformCore:message>
                  </platformCore:statusDetail>
               </platformCore:status>
            </getDataCenterUrlsReturn>
         </getDataCenterUrlsResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

getDeleted

You use the getDeleted operation to retrieve a list of deleted records. When you use this operation, you can filter by record type, by script ID, and by the date of record deletion. For each record that matches the filter criteria, the system returns the following data:

- The record’s type
- Its internal ID
- The date the record was deleted

The getDeleted operation can be useful when you want to synchronize information in a client application with NetSuite. For example, an Outlook client application plug-in maintains a list of contacts and synchronizes that list with NetSuite. The getDeleted operation can be used to determine contact deletions since the last synchronization with NetSuite.

Permissions

By default, only account administrators and users with the Full Access role have permission to use the getDeleted operation. However, you can grant permission to other roles. To use the getDeleted operation, the role has to have the Deleted Records permission (and the Web Services permission).

Users who have the Deleted Records permission can access results about any type of record that was deleted, even if they do not have permission to create or modify that record type. They also have permission to use the Deleted Record search type in the user interface, unless the Web Services Only option has been selected for the role.

You can give a role the Deleted Records permission by going to Setup > Users/Roles > Manage Roles. Go to the Permissions > Setup subtab, and add a line for the permission.

Usage Notes

Note the following:

- Not all record types support the getDeleted operation. For a complete list of supported record types for this operation, refer to the DeletedRecordType enumeration in the coreTypes xsd.
- Data about deleted records remains available indefinitely.
- The getDeleted response does not differentiate between records deleted through web services and through the UI.
- Even though it is not a search, the getDeleted operation respects the pageSize search preference.
- The getDeleted operation changed in 2015.1. If you are using the 2014.2 endpoint or earlier, refer to the Platform Guide for your WSDL version. To obtain an older version of the platform guide, see the help topic SuiteTalk Archives.

Request

The GetDeletedRequest type is used for the request.

Element Name	XSD Type	Notes
getDeletedFilter	GetDeletedFilter	See GetDeletedFilter.
pageIndex	xsd: int	Use this argument to specify which page of results to return in your response. If there is only one page of results, enter 1.

GetDeletedFilter

The GetDeletedFilter lets you retrieve data based on the date, record type, and script ID of the deleted records.

Element Name	XSD Type	Notes
deletedDate	SearchDateField	The date that the record was deleted. For example, you can search for records that were deleted since a particular date or between two dates.
type	SearchEnumMultiSelectField	The type of the record that was deleted. For a complete list of all of the values currently supported by the getDeleted operation, refer to the DeletedRecordType enumerations in the coreTypes XSD.
scriptID	SearchStringField	The scriptID identifying the specific custom record or custom transaction type. Use this field to narrow the results to custom records (or custom transactions) **only** of that particular type. If you do not specify a script ID, and rely only on the type field, the

Element Name	XSD Type	Notes
		operation returns details on all instances that were deleted, regardless of type.

Response

The DeletedRecord type is used for the response.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
totalRecords	xsd: int	The total number of records that meet the filter criteria.
pageSize	xsd: int	The total number of records listed on each page of the results.
totalPages	xsd: int	The total number of available pages.
pageIndex	xsd: int	The number identifying the current page of results.
deletedRecordList	deletedRecordList	A list of records that meet the filter criteria

Faults

This operation can throw any of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

The following sample shows how to retrieve details about deleted instances of a custom record type with the script ID customrecord1. It specifies that 10 results should be returned per page, and it includes logic to retrieve all pages.

Java

```
{
   c.setSearchPrefsPageSize(10);

   GetDeletedFilter filter = new GetDeletedFilter();
   filter.setType(new SearchEnumMultiSelectField(new String[] {DeletedRecordType._customRecord}
, SearchEnumMultiSelectFieldOperator.anyOf));
   filter.setScriptId(new SearchStringField("customrecord1", SearchStringFieldOperator.is));

   GetDeletedResult result;
      int i = 1;
      do
      {
         result = c.m_port.getDeleted(filter, i);
         if (i == 1)
         {
            System.out.println(String.format("Page Size : %d, Total Pages : %d, Total Records :
 %d", result.getPageSize(), result.getTotalPages(), result.getTotalRecords()));
         }
         processDeletedRecordList(result.getDeletedRecordList());
         System.out.println(String.format("Current Page Index : %d", result.getPageIndex()));
         i++;
      }
      while (i <= result.getTotalPages());
}
```

getItemAvailability

The getItemAvailability operation can be used to retrieve the inventory availability for a specific list of items.

You can filter the returned list using a lastQtyAvailableChange filter. If set, only items with quantity available changes recorded as of this date are returned.

If the Multi-Location Inventory feature is enabled, this operation returns results for all locations. For locations that do not have any items available, only location IDs and names are listed in results.

Warning: This operation supports up to 10,000 records. If this limit is exceeded, an error is returned.

Request

The GetItemAvailabilityRequest type is used for the request.

Element Name	XSD Type	Notes
itemAvailabilityFilter	ItemAvailabilityFilter	You can filter the returned itemAvailability using this filter.

ItemAvailabilityFilter

Element Name	XSD Type	Notes
item	RecordRefList	References an exiting item record in NetSuite.
lastQtyAvailableChange	dateTime	If set, only items with quantity available changes recorded as of the specified date are returned.

Response

The GetItemAvailabilityResult type is used for the response.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
itemAvailabilityList	List	Returns a list of available items.

ItemAvailabilityList

Element Name	XSD Type	Notes
item	RecordRef	References an existing item record.
lastQtyAvailableChange	dateTime	If set, only items with quantity available changes recorded as of this date are returned.
locationId	RecordRef	References a location in a user defined list at Lists > Accounting > Locations.
quantityOnHand	double	The number of units of an item in stock.
onHandValueMli	double	
reorderPoint	double	The stock level at which a new order for the item needs to be placed
preferredStockLevel	double	The preferred quantity of this item maintained in inventory for a specific location.
quantityOnOrder	double	The number of units of an item pending receipt from a vendor.
quantityCommitted	double	The number of units of an item reserved by unfulfilled sales orders.
quantityBackOrdered	double	The number of units of an item reserved by unfulfilled sales orders.
quantityAvailable	double	The number of units in stock that have not been committed to fulfill sales.

ItemAvailabilityFilter

Element Name	XSD Type	Notes
item	RecordRefList	
lastQtyAvailableChange	dateTime	

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request (for C# sample)

```
<soap:Body>
<platformMsgs:getItemAvailability>
   <platformMsgs:itemAvailabilityFilter>
      <platformCore:item>
         <platformCore:recordRef internalId="390" type="inventoryItem">
            <platformCore:name/>
         </platformCore:recordRef>
      </platformCore:item>
      <platformCore:lastQtyAvailableChange/>
   </platformMsgs:itemAvailabilityFilter>
</platformMsgs:getItemAvailability>
</soap:Body>
```

SOAP Response (for C# sample)

```
<soapenv:Body>
<getItemAvailabilityResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <getItemAvailabilityResult
      xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
      <status isSuccess="true"/>
      <itemAvailabilityList>
         <itemAvailability>
            <item internalId="390" type="inventoryItem">
               <name>testItem</name>
            </item>
            <locationId internalId="1" type="location">
               <name>East Coast</name>
            </locationId>
            <quantityOnHand>20.0</quantityOnHand>
            <onHandValueMli>0.0</onHandValueMli>
            <quantityCommitted>0.0</quantityCommitted>
            <quantityAvailable>20.0</quantityAvailable>
         </itemAvailability>
         <itemAvailability>
            <item internalId="390" type="inventoryItem">
               <name>testItem</name>
            </item>
            <locationId internalId="2" type="location">
```

```
                <name>West Coast</name>
             </locationId>
          </itemAvailability>
       </itemAvailabilityList>
    </getItemAvailabilityResult>
</getItemAvailabilityResponse>
</soapenv:Body>
```

C# Sample

```
private void getItemAvailability()
        {
            this.login(true);

            RecordRef item1 = new RecordRef();
            item1.internalId = "59";
            item1.type = RecordType.inventoryItem;
            item1.typeSpecified = true;

            RecordRef[] recordrefs = new RecordRef[1];
            recordrefs[0] = item1;

            ItemAvailabilityFilter filter = new ItemAvailabilityFilter();
            filter.item = recordrefs;
            GetItemAvailabilityResult res = _service.getItemAvailability(filter);

        }
```

Java Sample

```
public void getItemAvailability() throws RemoteException {
                this.login(true);

                RecordRef item1 = new RecordRef();
                item1.setInternalId("25");
                item1.setType(RecordType.inventoryItem);

                RecordRef item2 = new RecordRef();
                item2.setInternalId("76");
                item2.setType(RecordType.giftCertificateItem);

                RecordRef[] recordRefArray = new RecordRef[2];
                recordRefArray[0] = item1;
                recordRefArray[1] = item2;

                RecordRefList itemRefList = new RecordRefList();
                itemRefList.setRecordRef(recordRefArray);

                ItemAvailabilityFilter itemAvailability = new ItemAvailabilityFilter();
                itemAvailability.setItem(itemRefList);

                GetItemAvailabilityResult response = _port.getItemAvailability(itemAvailability
);
        }
```

getList

The getList operation is used to retrieve a list of one or more records by providing the unique ids that identify those records.

If there are multiple ids provided, they can either belong to the same record type or different record types. For example, it is possible to retrieve a customer and a contact within a single request using this operation.

If some of the provided ids are invalid, the request is still processed for the valid ids and the response will contain a warning that indicates that some of the ids were invalid.

Note: An asynchronous equivalent is available for this operation, **asyncGetList**. For information about asynchronous request processing, see Synchronous Versus Asynchronous Request Processing.

Request

The getListRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
recordRef	RecordRef	An array of recordRef objects that specify the ids of the records to be retrieved.

Response

The getListResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
recordList	Record[]	A list of records that correspond to the specified ids. The actual records returned need to be of a type that extends the abstract type Record.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault

- UnexpectedErrorFault

Sample Code

SOAP Request

In the following example, two records are retrieved — one customer record and one employee record. Note that you must provide the internal ID of the specify instance of the record and the record type for the getList.

```
<soap:Body>
<platformMsgs:getList>
   <platformMsgs:baseRef internalId="983" type="customer" xsi:type="platformCore:RecordRef"/>
   <platformMsgs:baseRef internalId="-5" type="employee" xsi:type="platformCore:RecordRef"/>
</platformMsgs:getList>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
<getListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<readResponseList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <readResponse>
      <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com
"/>
      <record internalId="983" xsi:type="ns2:Customer"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:ns2="urn:relationships_2017_1.lists.webservices.netsuite.com">
         <ns2:entityId>Shutter Fly</ns2:entityId>
         <ns2:isInactive>false</ns2:isInactive>
         <ns2:companyName>Shutter Fly, Inc</ns2:companyName>
         .
         ...[more fields]
         .
         <ns2:customFieldList>
            <ns6:customField internalId="265" scriptId="custentity_map"
            xsi:type="ns6:StringCustomFieldRef"
            xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.com">
                  <ns6:value>http://maps.google.com</ns6:value>
            </ns6:customField>
         </ns2:customFieldList>
      </record>
   </readResponse>
   <readResponse>
      <ns8:status isSuccess="true" xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com
"/>
      <record internalId="-5" xsi:type="ns9:Employee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:ns9="urn:employees_2017_1.lists.webservices.netsuite.com">
         <ns9:entityId>A Wolfe</ns9:entityId>
         <ns9:isInactive>false</ns9:isInactive>
         .
         ...[more fields]
         .
      </record>
   </readResponse>
</readResponseList>
</getListResponse>
 </soapenv:Body>
```

C#

```
private int getCustomerList()
{
   // This operation requires a valid session
   this.login( true );

   // Prompt for list of nsKeys and put in an array
   _out.write( "\nnsKeys for records to retrieved (separated by commas): " );
   String reqKeys = _out.readLn();
   string [] nsKeys = reqKeys.Split( new Char[] {','} );

   return getCustomerList( nsKeys, false );
}
```

Java

```
public int getCustomerList() throws RemoteException,
ExceededUsageLimitFault, UnexpectedErrorFault, InvalidSessionFault,
ExceededRecordCountFault {
   // This operation requires a valid session
   this.login(true);

   // Prompt for list of nsKeys and put in an array
   _console
   .write("\nnsKeys for records to retrieved (separated by commas): ");
   String reqKeys = _console.readLn();
   String[] nsKeys = reqKeys.split(",");

   return getCustomerList(nsKeys, false);
}
```

getPostingTransactionSummary

The getPostingTransactionSummary operation lets you retrieve a summary of the actual data that posted to the general ledger in an account. You can use available filters and fields to generate reports that are similar to what you see when you run financial reports such as a Trial Balance, Balance Sheet, or an Income Statement.

Note: For information about NetSuite financial reports and financial statements, see the following topics:

- Financial Reports

- Financial Statements Overview

The getPostingTransactionSummary operation returns the fields defined in PostingTransactionSummary. You can query by any filter defined in PostingTransactionSummaryFilter and group the results by any field defined in PostingTransactionSummaryField.

The first call to the operation returns the first page, total number of hits (totalRecords), and the number of pages. You can then retrieve subsequent pages by giving the page number.

Note: NetSuite caches the results after the first call to getPostingTransactionSummary as subsequent pages are being retrieved. The cache is reset if the session expires, or if you make another call to this operation with a page index of 1.

Also note the following:

- This operation can only be executed in a role that has the Financial Statements permission assigned. To enable this permission for a role, a NetSuite administrator must go to Setup > User Roles > Manage Roles, click the Reports tab, select **Financial Statments** from the Permission list, and then click Save.
- To search for null, specify -1. If the return is null, the element will be skipped (not -1).
- A maximum of 10,000 expressions is supported. A request with more than this number of expressions returns an error.
- For very large reports (for example, you have chosen all the columns), the query will take a very long time on first request, but subsequent requests will be fast. Make sure your timeout limit is set high.

Important: The information in the following paragraph pertains to **NetSuite OneWorld** accounts **only**.

The amounts returned from getPostingTransactionSummary are in the currency of the **subsidiary**, not the parent. If you want the amounts in the currency of the parent, you must programmatically apply the appropriate exchange rate. To obtain exchange rates, you must call getConsolidatedExchangeRate, which reads data from the Consolidated Exchange Rate table. In your code, you must then multiply amount values returned by getPostingTransactionSummary by the exchange rate values returned by getConsolidatedExchangeRate.

Although the NetSuite UI automatically consolidates all amounts, you must perform your own exchange rate calculations in web services.

Note: You may want to do planning in your local currency, in which case there is no need for exchange rate conversions.

Request

The GetPostingTransactionSummaryRequest type is used for the request.

Element Name	XSD Type	Notes
fields	PostingTransactionSummaryField	Specify how you want your data grouped.
filters	PostingTransactionSummaryFilter	Specify your filtering criteria.
pageIndex	xsd:int	Specify the page to be returned.

PostingTransactionSummaryField

Element Name	XSD Type	Notes
period	RecordRef	Specify to group data by period.
account	RecordRef	Specify to group data by account.
parentItem	RecordRef	Specify to group data by parent item.
item	RecordRef	Specify to group data by item.
customer	RecordRef	Specify to group data by customer.
department	RecordRef	Specify to group data by department.
class	RecordRef	Specify to group data by class.
location	RecordRef	Specify to group data by location.
subsidiary	RecordRef	Specify to group data by subsidiary.
book	RecordRef	If you use the Multi-Book Accounting feature, specify to group data by accounting book.

PostingTransactionSummaryFilter

Element Name	XSD Type	Notes
period	RecordRef	Filter your request by period.
account	RecordRef	Filter your request by account.
parentItem	RecordRef	Filter your request by parent item.
item	RecordRef	Filter your request by item.
customer	RecordRef	Filter your request by customer.
department	RecordRef	Filter your request by department.
class	RecordRef	Filter your request by class.
location	RecordRef	Filter your request by location.
subsidiary	RecordRef	Filter your request by subsidiary.
book	RecordRef	If you use the Multi-Book Accounting feature, filter your request by accounting book.

Response

The GetPostingTransactionSummaryResult type is used for the response.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
totalRecords	xsd:int	The total number of records for this search. Depending on the pageSize

Element Name	XSD Type	Notes
		value, some or all the records may be returned in this response.
pageSize	xsd:int	The page size for this search.
totalPages	xsd:int	The total number of pages that are part of this search.
pageIndex	xsd:int	The page index for the current set of results.
postingTransactionSummaryList	PostingTransactionSummaryList	Returns a list of available transaction summary results based on filters defined in your request.

PostingTransactionSummaryList

Element Name	XSD Type	Notes
postingTransactionSummary	PostingTransactionSummary	Returns a list of available transactions that are filtered by and grouped by values specified in PostingTransactionSummaryField and PostingTransactionSummaryFilter.

PostingTransactionSummary

Element Name	XSD Type	Notes
period	RecordRef	Returns a summary based on period.
account	RecordRef	Returns a summary based on account.
parentItem	RecordRef	Returns a summary based on parent item.
item	RecordRef	Returns a summary based on item.
customer	RecordRef	Returns a summary based on customer.
department	RecordRef	Returns a summary based on department.
class	RecordRef	Returns a summary based on class.
location	RecordRef	Returns a summary based on location.
subsidiary	RecordRef	Returns a summary based on subsidiary.
book	RecordRef	If you use the Multi-Book Accounting feature, specify to group data by accounting book.
amount	double	Returns a summary based on amount. ⚠ **Important:** This argument is required.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

Example 1

The sample provided shows how to get and print account values using the getPostingTransactionSummary operation.

Java

```
/* Make Record Ref out of an internalId */
   public static RecordRef mrr(String internalId)
   {
      RecordRef toRet = new RecordRef();
      toRet.setInternalId(internalId);
      return toRet;
   }

public void testPostingActivity() throws Exception
{
//c.setHttpPort(80);
c.login();
PostingTransactionSummaryField pagb = new PostingTransactionSummaryField();
pagb.setParentItem(Boolean.TRUE);

/*
pagb.set_class(Boolean.FALSE);
pagb.setItem(Boolean.FALSE);
pagb.setCustomer(Boolean.FALSE);
pagb.setLocation(Boolean.FALSE);
*/

PostingTransactionSummaryFilter paf = new PostingTransactionSummaryFilter();
//paf.setAccount(new RecordRefList(new RecordRef[]{mrr("5"),mrr("12")}));
long start = System.currentTimeMillis();
GetPostingTransactionSummaryResult res = c.getPort().getPostingTransactionSummary(pagb, paf, 1)
;

for (int i=0; i<10; i++)
{
if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getAccount() != null
)
System.out.println("Account:" + res.getPostingTransactionSummaryList().getPostingTransactionSum
mary(i).getAccount().getInternalId());
if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getPeriod() != null)
System.out.println("Period:" + res.getPostingTransactionSummaryList().getPostingTransactionSumm
ary(i).getPeriod().getInternalId());
if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getCustomer() != nul
l)
System.out.println("Customer:" + res.getPostingTransactionSummaryList().getPostingTransactionSu
mmary(i).getCustomer().getInternalId());
if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).get_class() != null)

System.out.println("_class:" + res.getPostingTransactionSummaryList().getPostingTransactionSumm
ary(i).get_class().getInternalId());
if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getDepartment() != n
ull)
System.out.println("Department:" + res.getPostingTransactionSummaryList().getPostingTransaction
Summary(i).getDepartment().getInternalId());
if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getLocation() != nul
l)
System.out.println("Location:" + res.getPostingTransactionSummaryList().getPostingTransactionSu
mmary(i).getLocation().getInternalId());
if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getSubsidiary() != n
ull)
System.out.println("Subsidiary:" + res.getPostingTransactionSummaryList().getPostingTransaction
Summary(i).getSubsidiary().getInternalId());
if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getItem() != null)
System.out.println("Item:" + res.getPostingTransactionSummaryList().getPostingTransactionSummar
y(i).getItem().getInternalId());
System.out.println("Amount:" + res.getPostingTransactionSummaryList().getPostingTransactionSumm
ary(i).getAmount());
}
for (int i=res.getPageIndex(); i<res.getTotalPages(); i++)
{
System.err.println("Elapsed Time : " + (System.currentTimeMillis() - start));
res = c.getPort().getPostingTransactionSummary(pagb,paf,i+1);
}
System.err.println("Elapsed Time : " + (System.currentTimeMillis() - start));
GetPostingTransactionSummaryResult p = c.getPort().getPostingTransactionSummary(pagb, paf, res.
getTotalPages()+2);
assertFalse(p.getStatus().isIsSuccess());
}
```

Example 2

The following SOAP and Java show how to get the posting activity for posting period 109. The sample prints the unconsolidated and consolidated amounts in the parent subsidiary by child subsidiary and account – for example, if subsidiary **2** is in Euros and subsidiary **1**, the parent company, is in dollars, show the amount in Euros and the amount in dollars for that account in period 109.

SOAP Request

```
  <soapenv:Body>
         <getPostingTransactionSummary xmlns="urn:messages_2017_1.platform.webservices.netsuite
.com">
            <fields>
               <ns6:period xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.com">true</
```

```
ns6:period>
               <ns7:account xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">true<
/ns7:account>
               <ns8:subsidiary xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com">tr
ue<
         ns8:subsidiary>
            </fields>
            <filters>
               <ns9:period xmlns:ns9="urn:core_2017_1.platform.webservices.netsuite.com">
                  <ns9:recordRef internalId="109"/>
               </ns9:period>
            </filters>
            <pageIndex>1</pageIndex>
         </getPostingTransactionSummary>
      </soapenv:Body>
```

SOAP Reponse

```
<soapenv:Body>
         <getPostingTransactionSummaryResponse
      xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <platformCore:getPostingTransactionSummaryResult
      xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
               <platformCore:status isSuccess="true"/>
               <platformCore:totalRecords>19</platformCore:totalRecords>
               <platformCore:pageSize>1000</platformCore:pageSize>
               <platformCore:totalPages>0</platformCore:totalPages>
               <platformCore:pageIndex>1</platformCore:pageIndex>
               <platformCore:postingTransactionSummaryList>
                  <platformCore:postingTransactionSummary>
                     <platformCore:period internalId="109"/>
                     <platformCore:account internalId="2"/>
                     <platformCore:subsidiary internalId="1"/>
                     <platformCore:amount>500.0</platformCore:amount>
                  </platformCore:postingTransactionSummary>
                  <platformCore:postingTransactionSummary>
                     <platformCore:period internalId="109"/>
                     <platformCore:account internalId="6"/>
                     <platformCore:subsidiary internalId="1"/>
                     <platformCore:amount>19.0</platformCore:amount>
                  </platformCore:postingTransactionSummary>
                  <platformCore:postingTransactionSummary>
                     <platformCore:period internalId="109"/>
                     <platformCore:account internalId="6"/>
                     <platformCore:subsidiary internalId="2"/>
                     <platformCore:amount>5.0</platformCore:amount>
                  </platformCore:postingTransactionSummary>
                  <platformCore:postingTransactionSummary>
                     <platformCore:period internalId="109"/>
                     <platformCore:account internalId="6"/>
                     <platformCore:subsidiary internalId="4"/>
                     <platformCore:amount>6.0</platformCore:amount>
                  </platformCore:postingTransactionSummary>
                  <platformCore:postingTransactionSummary>
```

```
        <platformCore:period internalId="109"/>
        <platformCore:account internalId="6"/>
        <platformCore:subsidiary internalId="8"/>
        <platformCore:amount>-3.0</platformCore:amount>
    </platformCore:postingTransactionSummary>
    <platformCore:postingTransactionSummary>
        <platformCore:period internalId="109"/>
        <platformCore:account internalId="16"/>
        <platformCore:subsidiary internalId="6"/>
        <platformCore:amount>-4.0</platformCore:amount>
    </platformCore:postingTransactionSummary>
    <platformCore:postingTransactionSummary>
        <platformCore:period internalId="109"/>
        <platformCore:account internalId="24"/>
        <platformCore:subsidiary internalId="8"/>
        <platformCore:amount>-100.0</platformCore:amount>
    </platformCore:postingTransactionSummary>
    <platformCore:postingTransactionSummary>
        <platformCore:period internalId="109"/>
        <platformCore:account internalId="58"/>
        <platformCore:subsidiary internalId="8"/>
        <platformCore:amount>100.0</platformCore:amount>
    </platformCore:postingTransactionSummary>
    <platformCore:postingTransactionSummary>
        <platformCore:period internalId="109"/>
        <platformCore:account internalId="112"/>
        <platformCore:subsidiary internalId="1"/>
        <platformCore:amount>-496.0</platformCore:amount>
    </platformCore:postingTransactionSummary>
    <platformCore:postingTransactionSummary>
        <platformCore:period internalId="109"/>
        <platformCore:account internalId="112"/>
        <platformCore:subsidiary internalId="5"/>
        <platformCore:amount>-1.0</platformCore:amount>
    </platformCore:postingTransactionSummary>
    <platformCore:postingTransactionSummary>
        <platformCore:period internalId="109"/>
        <platformCore:account internalId="139"/>
        <platformCore:subsidiary internalId="4"/>
        <platformCore:amount>-9.0</platformCore:amount>
    </platformCore:postingTransactionSummary>
    <platformCore:postingTransactionSummary>
        <platformCore:period internalId="109"/>
        <platformCore:account internalId="142"/>
        <platformCore:subsidiary internalId="8"/>
        <platformCore:amount>3.0</platformCore:amount>
    </platformCore:postingTransactionSummary>
    <platformCore:postingTransactionSummary>
        <platformCore:period internalId="109"/>
        <platformCore:account internalId="145"/>
        <platformCore:subsidiary internalId="5"/>
        <platformCore:amount>1.0</platformCore:amount>
    </platformCore:postingTransactionSummary>
    <platformCore:postingTransactionSummary>
        <platformCore:period internalId="109"/>
```

```
                  <platformCore:account internalId="148"/>
                  <platformCore:subsidiary internalId="2"/>
                  <platformCore:amount>-5.0</platformCore:amount>
               </platformCore:postingTransactionSummary>
               <platformCore:postingTransactionSummary>
                  <platformCore:period internalId="109"/>
                  <platformCore:account internalId="153"/>
                  <platformCore:subsidiary internalId="6"/>
                  <platformCore:amount>4.0</platformCore:amount>
               </platformCore:postingTransactionSummary>
               <platformCore:postingTransactionSummary>
                  <platformCore:period internalId="109"/>
                  <platformCore:account internalId="155"/>
                  <platformCore:subsidiary internalId="1"/>
                  <platformCore:amount>-23.0</platformCore:amount>
               </platformCore:postingTransactionSummary>
               <platformCore:postingTransactionSummary>
                  <platformCore:period internalId="109"/>
                  <platformCore:account internalId="175"/>
                  <platformCore:subsidiary internalId="4"/>
                  <platformCore:amount>3.0</platformCore:amount>
               </platformCore:postingTransactionSummary>
               <platformCore:postingTransactionSummary>
                  <platformCore:period internalId="109"/>
                  <platformCore:account internalId="186"/>
                  <platformCore:subsidiary internalId="8"/>
                  <platformCore:amount>0.0</platformCore:amount>
               </platformCore:postingTransactionSummary>
               <platformCore:postingTransactionSummary>
                  <platformCore:period internalId="109"/>
                  <platformCore:account internalId="187"/>
                  <platformCore:subsidiary internalId="8"/>
                  <platformCore:amount>0.0</platformCore:amount>
               </platformCore:postingTransactionSummary>
            </platformCore:postingTransactionSummaryList>
         </platformCore:getPostingTransactionSummaryResult>
      </getPostingTransactionSummaryResponse>
   </soapenv:Body>
```

Java

```
/* Make Record Ref out of an internalId */
   public static RecordRef mrr(String internalId)
   {
      RecordRef toRet = new RecordRef();
      toRet.setInternalId(internalId);
      return toRet;
   }

   public void testPostingWorkflow() throws Exception
   {
      c.setCredentials(CRED_DB96_SIC);
      c.useRequestLevelCredentials();
      // Show and group by subsidiary, period and account.
      // These are the very basic columns. If you do not include account,
      // all amounts will be 0.
      PostingTransactionSummaryField pagb = new PostingTransactionSummaryField();
      pagb.setSubsidiary(Boolean.TRUE);
      pagb.setPeriod(Boolean.TRUE);
      pagb.setAccount(Boolean.TRUE);

      PostingTransactionSummaryFilter paf = new PostingTransactionSummaryFilter();
      paf.setPeriod(new RecordRefList(new RecordRef[]{mrr("109")}));
      GetPostingTransactionSummaryResult gestalt =
         c.getPort().getPostingTransactionSummary(pagb,paf,1);
      ConsolidatedExchangeRateFilter f = new ConsolidatedExchangeRateFilter();
      f.setPeriod(mrr("109"));
      f.setToSubsidiary(mrr("1"));
      GetConsolidatedExchangeRateResult cerr = c.getPort().getConsolidatedExchangeRate(f);

      for (PostingTransactionSummary unConsolidated :
         gestalt.getPostingTransactionSummaryList().getPostingTransactionSummary())
      {
         ConsolidatedExchangeRate rate = null;
         for (ConsolidatedExchangeRate testRate :
            cerr.getConsolidatedExchangeRateList().getConsolidatedExchangeRate())
         {
            if
               (testRate.getFromSubsidiary().getInternalId().equals
               (unConsolidated.getSubsidiary().getInternalId()))
            {
               rate = testRate;
               break;
            }
         }
         if (rate == null)
            continue;  // the Target Subsidiary will not have a conversion rate to itself.
         System.out.println("\n\nFor Subsidiary: " + unConsolidated.getSubsidiary().getInternal
Id()  +
            " to Consolidated Parent (1).\nPeriod: " + unConsolidated.getPeriod().getInternalId
() +
                  "\nAccount Id: " + unConsolidated.getAccount().getInternalId() +
                  "\nUnconsolidated Amount is: " + unConsolidated.getAmount() +
                  "\nConsolidated Amounts are (Avg/Historical/Current): "
                     + (unConsolidated.getAmount()*rate.getAverageRate()) + " / " +
                     (unConsolidated.getAmount()*rate.getHistoricalRate()) + " / " +
                     (unConsolidated.getAmount()*rate.getCurrentRate()));
      }
   }
```

getSavedSearch

This operation allows users to retrieve a list of existing saved search IDs on a per-record-type basis (for example, all saved search IDs for every Customer saved search). Note that after you retrieve the list of saved search IDs, you may need to look in the NetSuite UI to see the criteria defined for the saved search. To navigate to the list of saved searches in the NetSuite UI, go to Lists > Search > Saved Searches.

This API takes a search record type as a request argument and returns a list of record references of the saved search. For each saved search, the response includes the internalId, scriptId, and name (if you are using the 2013.2 WSDL or later — with previous WSDLs, the scriptId is not returned).

For use cases explaining why you would want to get a list of saved search IDs and then reference a specific ID in your code, see Reference Existing Saved Searches.

Note: There is no async equivalent for this operation.

Request

The GetSavedSearchRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
record	GetSavedSearchRecord	

Response

The GetSavedSearchResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
totalRecords	xsd:int	The total number of records for this search. Depending on the pageSize value, some or all the records may be returned in this response
recordRefList	Record[]	A list of records that correspond to the specified ids. The actual records returned need to be of a type that extends the abstract type Record.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

C#

```
GetSavedSearchRecord record = new GetSavedSearchRecord();
record.searchTypeSpecified = true;
record.searchType = SearchRecordType.transaction;

_service.getSavedSearch(record);
;
```

Java

```
public void getSavedSearches() throws RemoteException{
  this.login(true);

  GetSavedSearchRecord record = new GetSavedSearchRecord();
  record.setSearchType(SearchRecordType.transaction);

  GetSavedSearchResult result = _port.getSavedSearch(record);
}
```

SOAP Request

```
   <soap:Body>
        <getSavedSearch xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record searchType="transaction"/>
        </getSavedSearch>
    </soap:Body>
```

SOAP Response

```
<soapenv:Body>
   <getSavedSearchResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <platformCore:getSavedSearchResult xmlns:platformCore="urn:core_2017_1.platform.webservic
es.netsuite.com">
         <platformCore:status isSuccess="true"/>
      <platformCore:totalRecords>5</platformCore:totalRecords>
      <platformCore:recordRefList>
         <platformCore:recordRef xsi:type="platformCore:CustomizationRef" internalId="26" scrip
tId="customsearch26">
            <platformCore:name>Custom Transaction Search</platformCore:name>
          </platformCore:recordRef>
            <platformCore:recordRef xsi:type="platformCore:CustomizationRef" internalId="27" sc
riptId="customsearch27">
               <platformCore:name>CC Transaction Search</platformCore:name>
            </platformCore:recordRef>
            <platformCore:recordRef xsi:type="platformCore:CustomizationRef" internalId="42" sc
```

```
riptId="customsearch42">
                <platformCore:name>Returned Items</platformCore:name>
         </platformCore:recordRef>
         <platformCore:recordRef xsi:type="platformCore:CustomizationRef" internalId="44" scrip
tId="customsearch44">
               <platformCore:name>Ordered More than One Item</platformCore:name>
         </platformCore:recordRef>
            <platformCore:recordRef xsi:type="platformCore:CustomizationRef" internalId="46" sc
riptId="customsearch46">
               <platformCore:name>Spending Account Average</platformCore:name>
            </platformCore:recordRef>
         </platformCore:recordRefList>
      </platformCore:getSavedSearchResult>
   </getSavedSearchResponse>
</soapenv:Body>
```

getSelectValue

Use the getSelectValue operation to retrieve valid select options for a particular RecordRef, CustomRecordRef, or enumerated static field. This is useful if you are writing an application UI that needs to mimic NetSuite UI logic, if the referenced record type is not yet exposed in SuiteTalk, or when the logged-in user's role does not have permission to the instances of the referenced record type. A call to getSelectValue may return different results for the same field for different roles.

The getSelectValue operation can be used on standard body fields and custom body fields. It can also be used on sublist fields that appear on both standard and custom records.

These topics describe each aspect of the operation:

- GetSelectValue Overview
- Paginating Select Values
- Filtering Select Value Text
- Getting Dependent Select Values
- Sample Code

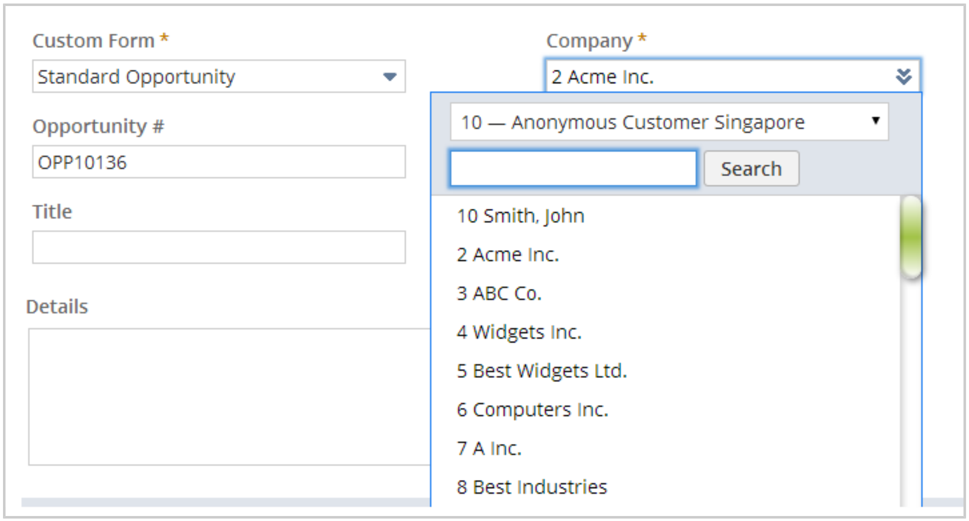
Important: If you reference a field or a select value that is renamed in future versions of NetSuite, your requests will still be handled, however, a warning will be returned. Also, when working with this operation be aware of any special permissions applied to a field. For example, a permission error will be thrown if you attempt to get select values on a field that has been disabled on a form.

Note: The getSelectValue operation will not return the following values: "" , (blank), -1,

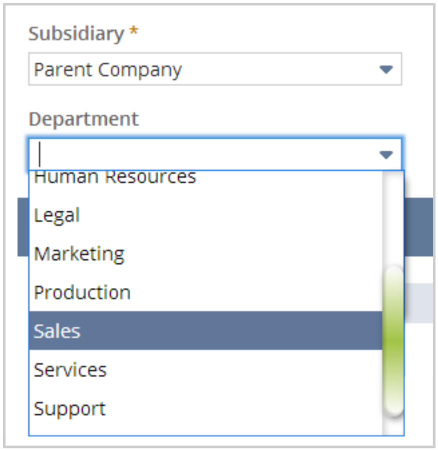
-New-, -2, -Custom-.

GetSelectValue Overview

The following figure shows a RecordRef field (Company) as displayed in the user interface, along with a list of select values for this field. You can use getSelectValue to return the entire list of values or a subset of values.



In accounts where there are levels of field dependencies, such as OneWorld accounts, you can use getSelectValue to get select values for field “B” based on the value of field “A”. The following screenshot shows an employee record in a OneWorld account. The select values associated with the Department field are based on the value specified in the Subsidiary field. In this scenario, you can use getSelectValue to get the values on the Department field by specifying the internal ID of its filterBy (“master”) field, which is Subsidiary (internal ID —**subsidiary**). For more information, see Getting Dependent Select Values.



Running getSelectValue against records that are created from other records may result in an INSUFFICIENT_PERMISSION error.

Paginating Select Values

The first call to getSelectValue returns the total number of select values for the specified field. If you choose, you can return a subset of those values by specifying a pageIndex number in the request.

```
<complexType name="getSelectValueRequest">
       <sequence>
           <element name="fieldDescription" type="platformCore:GetSelectValueFieldDescription"

     minOccurs="1" maxOccurs="1"/>
```

```
      <element name=" pageIndex " type="xsd:int" minOccurs="1" maxOccurs="1"/>
      </sequence>
    </complexType>
```

Note: To define a page size, set the pageSize element in the SearchPreference type (see . The value must be greater than 10 and less than the system-defined maximum of 1000. If the number of select values exceeds the page size, the remaining results must be retrieved in subsequent calls getSelectValue with a new pageIndex value.

Filtering Select Value Text

To help end users pick from a long list of select values, you can use the **contains, startsWith**, or **is** search operators to filter the results returned by getSelectValue. The **filter** element in the GetSelectValueFieldDescription object lets you set the operator type in your request.

```
<complexType name=" GetSelectValueFieldDescription ">
   <sequence>
      <element name="recordType" type="platformCoreTyp:RecordType" minOccurs="0" maxOccurs="1"/
>
      <element name="customRecordType" type="platformCore:RecordRef" minOccurs="0"
      maxOccurs="1"/>
                  <element name="sublist" type="xsd:string" minOccurs="0" maxOccurs="1"/>
         <element name="field" type="xsd:string" minOccurs="1" maxOccurs="1"/>
         <element name="customForm" type="platformCore:RecordRef" minOccurs="0" maxOccurs="1"/>

         <element name=" filter " type="platformCore: GetSelectValueFilter " minOccurs="0" max0
ccurs="1"/>
         <element name="filterByValueList" type="platformCore:GetSelectFilterByFieldValueList"
      minOccurs="0" maxOccurs="1"/>
        </sequence>
</complexType>
```

Getting Dependent Select Values

The getSelectValue operation can also be used to get select values that are available on the condition of other field values. For example, in OneWorld accounts the values that appear in many dropdown fields are based on the values specified for either Customer or Subsidiary. As another example, on the Item sublist of a Sales Order, the values for the Tax Code field depend on both customer and item values.

In your getSelectValue call, you will use the **GetSelectFilterByFieldValueList** and **GetSelectFilterByFieldValue** objects to specify the filterBy field of a dependent field. (The dependent field is the field you want to get the values for. The filterBy field is the field that controls which values are available for the dependent field.)

```
<complexType name=" GetSelectFilterByFieldValueList ">
      <sequence>
        <element name="filterBy" type="platformCore:GetSelectFilterByFieldValue" minOccurs="1"
         maxOccurs="unbounded"/>
      </sequence>
   </complexType>
   <complexType name=" GetSelectFilterByFieldValue ">
```

```
    <sequence>
      <element name=" sublist " type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <element name=" field " type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <element name=" internalId " type="xsd:string" minOccurs="1" maxOccurs="1"/>
    </sequence>
</complexType>
```

The sublist, field, and internalId arguments will contain:

- the name of the sublist the filterBy field appears on (*if* it appears on a sublist)
- the schema name of the filterBy field (for example, “entity”)
- the internalId of the specific entity record (for example, “87”)

A getSelectValue call for a dependent field with no filterBy (“master”) field specified returns 0 records. This type of call also returns a warning that notes the filterBy field, so the call can be corrected.

Important: Currently there is no programmatic way to discover what the filterBy field is for another field. Note, however, on transaction records the Customer (**entity**) field is always the filterBy field for any dependent RecordRef field. For all other record types, you must use the UI to see which field is the filterBy field.

Request

The getSelectValueRequest type is used for the request. It contains the following elements.

Element Name	XSD Type	Notes
fieldDescription	GetSelectValueFieldDescription	Use to specify all characteristics of the field containing the select values. For example, depending on the field and the values you want returned, you will specify the names or internalIds for the record type, sublist, and field. You may also specify filtering criteria to refine the select options returned in the response.
pageIndex	xsd: int	For select values that span multiple pages, use this argument to specify which page to return in your response.

GetSelectValueFieldDescription

Element Name	XSD Type	Notes
recordType	RecordType	Specify a record defined in coreTypes.xsd
customRecordType	RecordRef	If you are getting select values for a field appearing on a custom record, specify the internal or external ID of the custom record, as well as the custom record type.
sublist	xsd:string	If getting select values for a field on a sublist, you must specify the sublist name (as it is defined in the schema). Sublist names are those names that appear toward the bottom of a record schema and are appended with

Element Name	XSD Type	Notes
		List, for example itemList, saleTeamList, timeList.
field	xsd:string	Specify a field name, as defined in the schema for that record. The field value can represent either a body field or a sublist field.
customForm	RecordRef	If the RecordRef or CustomRecordRef field is associated with a custom form, specify the internal or external ID of the custom form, as well as the custom form type.
filter	GetSelectValueFilter	If you choose, you can filter select options using the **contains, is**, or **startsWith** operator. Use any of these operators to return a subset of the options associated with the RecordRef or CustomRecordRef field. For example, to get a specific list of customers on an opportunity record, you can search for "Adam" with the **contains** operator to get only customers whose name contains Adam. See also Filtering Select Value Text.
filterByValueList	GetSelectFilterByFieldValue List	This will contain a reference to the filterBy field (or fields) in which you specify the following: ■ **sublist** – If the filterBy field is on a sublist, you will specify the name of the sublist containing this field. Sublist names are appended with List (for example item **List**). ■ **field** – The name of the filterBy field (for example, *entity* or *subsidiary*). ■ **internalId** – The internalId value of the filterBy field (for example, "87" or "112"). If the filterBy field on an Opportunity record is Customer, you would specify: sublist = null field = "entity" internalId = "87" (87 being the internalId of a specific customer record, such as the Abe Simpson customer record) See also Getting Dependent Select Values.

Response

The getSelectValueResponse type is used for the response. It contains the following elements.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.

Element Name	XSD Type	Notes
totalRecords	xsd:int	The total number of record references for this search. Depending on the pageSize value, some or all the references may be returned in the response.
totalPages	xsd:int	The total number of pages for this search. Depending on the pageSize value, some or all the pages may be returned in this response.
baseRefList	BaseRef[]	An array of baseRefs that references existing NetSuite records, including custom records. These baseRefs represent the valid values available for the current field.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.o
rg
soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001
XMLSchema-instance">
 <soapenv:Header>
  <ns1:passport soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next" soapenv:mustUndersta
nd="0"
     xmlns:ns1="urn:messages_2017_1.platform.webservices.netsuite.com">
  <ns2:email xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">kwolfe@netsuite.com<

    ns2:email>
 <ns3:password xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">mypassword<
   ns3:password>
<ns4:account xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com">000071</ns4:account
<ns5:role internalId="3" xmlns:ns5="urn:core_2017_1.platform.webservices.netsuite.com"/>
  </ns1:passport>
 </soapenv:Header>
 <soapenv:Body>
  <getSelectValue xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <fieldDescription>
    <ns6:recordType xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.com">salesOrder<
       ns6:recordType>
```

```
<ns7:sublist xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">itemList</ns7:sublis
t>
<ns8:field xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com">item</ns8:field>
<ns9:filterByValueList xmlns:ns9="urn:core_2017_1.platform.webservices.netsuite.com">
     <ns9:filterBy>
      <ns9:field>entity</ns9:field>
      <ns9:internalId>8</ns9:internalId>
     </ns9:filterBy>
    </ns9:filterByValueList>
   </fieldDescription>
   <pageIndex>1</pageIndex>
  </getSelectValue>
 </soapenv:Body>
</soapenv:Envelope>
```

Java

This sample shows how get select values for the Item field that appears on the Item sublist of a Sales Order record.

```
GetSelectFilterByFieldValueList myFilterByList = new GetSelectFilterByFieldValueList(new
   GetSelectFilterByFieldValue[]{new GetSelectFilterByFieldValue(null,"entity","8")});

GetSelectValueFieldDescription myGSVField = new GetSelectValueFieldDescription(RecordType.sales
Order,
         null, "itemList", "item", null, null, myFilterByList);

BaseRef[] br = c.getSelectValue(myGSVField);
```

getServerTime

This operation takes no arguments and returns the NetSuite server time in GMT, regardless of a user's time zone. Developers do not have to rely on client time when writing synchronization procedures because the client time may not be in synch with the NetSuite server time.

If you choose, you can write client code that takes the GMT returned by NetSuite and convert the time to your local time zone. The format of the dateTime value returned by getServerTime contains integer-valued year, month, day, hour and minute properties, a decimal-valued second property and a boolean time-zoned property (timestamp) – for example, 2005-09-21T15:24:00.000-07:00, where 2005-09-21 is the date, 15:24:00.000 is the time and -07:00 is your own time zone offset, if you choose to set one.

Any user with an active session can call getServerTime. There are no explicit permissions for this operation. For example, if you need to synchronize customer data with NetSuite, you can call getServerTime to initialize your synchronization process. To synchronize new or changed customers records, you can search customers and filter by **lastModifiedDate** using the value returned in getServerTime. To synchronize deleted customer records, you can call the getDeleted operation and use the value returned in getServerTime. This makes your synchronization process independent of your client time,which may not be in sync with the NetSuite server time and hence the timestamp we use to track changes.

Request

The GetServerTimeRequest type is used for the request. It takes no arguments.

Response

The GetServerTimeResponse type is used for the response. It contains the following elements.

Element Name	XSD Type	Notes
response	GetServerTimeResult	See GetServerTimeResult, below.

GetServerTimeResult

The GetServerTimeResult type contains the following elements.

Element Name	XSD Type	Notes
status	Status	The status for this request. All applicable errors or warnings will be listed within this type, which is defined in the platformCore XSD.
serverTime	dateTime	dateTime value returned is in GMT.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- UnexpectedErrorFault

Sample Code

Java

```
public void testGetServerTime() throws Exception
{
   c.login();

   GetServerTimeResult rs = c.getPort().getServerTime();
   System.out.println("Welcome to NetSuite. At the sound of the tone the NetSuite time will be
: " + new
   SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(rs.getServerTime().getTime()));

   System.out.println("This compares with a client time of " + new SimpleDateFormat("yyyy-MM-dd

   HH:mm:ss").format(Calendar.getInstance().getTime() ));
   System.out.println("This represents a skew of  "  + (Calendar.getInstance().getTimeInMillis(
)
   rs.getServerTime().getTimeInMillis() )/1000 + " seconds from netsuite
   (where positive time means the client clock is comparatively fast).");
}
```

initialize / initializeList

Use the initialize operation to emulate the UI workflow by prepopulating fields on transaction line items with values from a related record. Your web services application can then modify only the values it needs to before submitting the record.

For example, in the UI clicking Bill from a Sales Order record loads an Invoice record where fields are populated with values from the Sales Order. When loading an invoice record in web services, you can reference the related Sales Order record to initialize fields with values from that sales order.

Note: An asynchronous equivalent is available for the initializeList operation, **asyncInitializeList**. For information about asynchronous request processing, see Synchronous Versus Asynchronous Request Processing.

The following table lists all the transaction types that can be used with the initialize operation and the valid reference types they can use.

(Target Record) Transaction Type	Initialize Reference	Notes
Assembly Unbuild	Assembly Build	
Cash Refund	Cash Sale	
Cash Refund	Return Authorization	
Cash Sale	Customer	Populates Billable tabs
Cash Sale	Estimate	
Cash Sale	Opportunity	
Cash Sale	Sales Order	
Credit Memo	Customer	
Credit Memo	Invoice	
Credit Memo	Return Authorization	
Customer Payment	Customer	Use the arAccount parameter to specify the accounts receivable (AR) account in a Customer to Customer Payment initialization. (This parameter is define in InitializeAuxRefType.)
Customer Payment	Invoice	
Customer Refund	Credit Memo	
Customer Refund	Customer	
Deposit Application	Customer Deposit	
Estimate	Opportunity	
Invoice	Customer	Defaults billable time, expense, and items
Invoice	Estimate	
Invoice	Opportunity	
Invoice	Sales Order	Defaults the line items as well as billable time, expense, and items
Item Fulfillment	Intercompany Transfer Order	

(Target Record) Transaction Type	Initialize Reference	Notes
Item Fulfillment	Sales Order	Defaults the line items for fulfillment
Item Fulfillment	Transfer Order	The line field on a transfer order does not have to correspond to the orderLine field on a newly created item fulfillment.
Item Fulfillment	Vendor Return Authorization	
Item Receipt	Intercompany Transfer Order	
Item Receipt	Purchase Order	Initializing an item receipt from a **dropship** purchase order is not supported. The recommended workflow is to initialize an item fulfillment from the sales order that has the dropship item. The accounting impact will be the same and more desirable because inventory levels will not be affected.
Item Receipt	Return Authorization	
Item Receipt	Transfer Order	
Purchase Order	Requisition	
Return Authorization	Cash Sale	
Return Authorization	Invoice	
Return Authorization	Sales Order	
Sales Order	Estimate	
Sales Order	Opportunity	
Vendor Bill	Purchase Order	You can initialize or link a single purchase order to a vendor bill, or a list of purchase orders to a vendor bill. Note, however, when linking multiple purchase orders to a single vendor bill, all purchase orders must be for the same vendor. For more information, see the help topic Linking Purchase Orders to a Vendor Bill.
Vendor Bill	Vendor	Use the apAccount parameter to specify the accounts payable (AP) account in a Vendor to Vendor Bill initialization. (This parameter is define in InitializeAuxRefType.)
Vendor Credit	Vendor	
Vendor Credit	Vendor Bill	
Vendor Credit	Vendor Return Authorization	
Vendor Payment	Employee	
Vendor Payment	Vendor	
Vendor Payment	Vendor Bill	
Vendor Return Authorization	Vendor	

(Target Record) Transaction Type	Initialize Reference	Notes
Vendor Return Authorization	Vendor Bill	
Work Order Close	Work Order	The work order must be configured to use WIP.
Work Order Completion	Work Order	The work order must be configured to use WIP.
Work Order Issue	Work Order	The work order must be configured to use WIP.

Ignore Read-Only Preference

To submit an initialized record without having to remove read-only fields populated during the initialization, set the Ignore Read-Only header preference to TRUE.

Important: When this preference is set to TRUE, read-only fields are ignored during the web services request. The fields still cannot be set.

Request

The InitializeRequest type is used for this request. It contains the following fields:

Element Name	XSD Type	Notes
initializeRecord	InitializeRecord	

The InitializeRecord type takes the following fields:

- type
- reference
- auxReference
- referenceList

Response

The InitializeResponse type is used for the response. It contains the following fields:

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
record	Record	The record type of the resulting initialization.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault

- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request (initialize)

```
<soapenv:Body>
<platformMsgs:initialize xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="h
ttp://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:pla
tformCoreTyp="urn:types.core_2017_1.platform.webservices.netsuite.com" xmlns:platformCore="urn:
core_2017_1.platform.webservices.netsuite.com" xmlns:platformMsgs="urn:messages_2017_1.platform
.webservices.netsuite.com">
   <platformMsgs:initializeRecord>
      <platformCore:type>invoice</platformCore:type>
      <platformCore:reference internalId="1513" type="salesOrder">
         <platformCore:name>1511</platformCore:name>
      </platformCore:reference>
   </platformMsgs:initializeRecord>
</platformMsgs:initialize>
</soapenv:Body>
```

SOAP Request (initializeList)

```
<initializeList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <initializeRecord>
    <ns1:type
       xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com">customerPayment
    </ns1:type>
    <ns2:reference internalId="176" type="customer"
         xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
  </initializeRecord>
  <initializeRecord>
    <ns3:type xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">
         invoice</ns3:type>
        <ns4:reference internalId="176" type="customer"
             xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com"/>
   </initializeRecord>
</initializeList>
```

Java (initializeList)

```
InitializeRef iRef1 = new InitializeRef();
iRef1.setInternalId("176");
iRef1.setType(InitializeRefType.customer);

InitializeRecord ir1 = new InitializeRecord();
ir1.setReference(iRef1);
```

```
ir1.setType(InitializeType.customerPayment);

InitializeRecord ir2 = new InitializeRecord();
ir2.setReference(iRef1);
ir2.setType(InitializeType.invoice);

sessMgr.getPort().getNetSuitePortTypePort().initializeList(new InitializeRecord[]{ir1, ir2});
```

C# (initialize)

```
private void Initialize()
        {
            this.login(true);

            InitializeRef ref1 = new InitializeRef();
            ref1.type = InitializeRefType.salesOrder;

               //internal id of the sales order to be converted to cash sale
            ref1.internalId = "792";
            ref1.typeSpecified = true;

            InitializeRecord rec = new InitializeRecord();
            rec.type = InitializeType.cashSale;
            rec.reference = ref1;

            ReadResponse read1 = _service.initialize(rec);
                }
```

login

The login operation is used to authenticate a user and start a new web services session in NetSuite. The login operation is similar to the login for the UI. This operation provides a passport that includes a username, password, account and role. On success, the NetSuite server sets a cookie and establishes a session.

Important: Users can also authenticate to NetSuite by providing their user credentials in the SOAP header of each request; they do not need to invoke the login operation. With user credentials provided in each request, the need for session management and separate logins is eliminated. For more information, see Request-Level Credentials.

All available web services operations require that the user first be logged in. After successfully completed, a login creates a session that allows subsequent operations to be performed without having to log in again until the session expires or the logout operation is invoked. If the session times out, the next operation fails. Web services requests initiated through a client application must have the ability to execute the login operation when a session has timed out and then submit the original request again.

Note: For information on session management, refer to Session Management for Web Services.

If an operation is executed before a login is performed, it fails and the InvalidSessionFault is returned. Also note that after login, only one web services request may be in flight at a time for a specific session. Any attempt to violate this will result in a SOAP fault.

NetSuite web services will use the last logged-in role when users do not specify a role in the login request and no default role is set. This makes the web services login behavior consistent with the UI login behavior. Partner applications that rely on a specific role should be careful to specify that role in the login request, otherwise their users might be logged in with a role that is not appropriate for that application.

The login operation also verifies that the specified account has the Web Services feature enabled. If the feature is not enabled in your account, the SOAP fault InvalidCredentialsFault is returned with a code of WEB_SERVICES_NOT_ENABLED. (See Enabling the Web Services Feature for steps on enabling this feature.)

Request

The LoginRequest type is used for this request. It contains the following fields:

Element Name	XSD Type	Notes
passport	Passport	Contains all the required credentials including username, password, account and role to authenticate the user and create a new session.

The Passport type includes the following elements:

- email
- password
- account
- role

Note: You can confirm your accountID in the NetSuite UI. As administrator, go to Support > Customer Service > Contact Support by Phone. Your account number is displayed in a pop-up box. Also, Role is not a required parameter in the WS login. However, if you do not specify a role, the user's default role must have WS permissions.

Response

The LoginResponse type is used for the response. This references the SessionResponse type, which includes the status and wsRoleList elements.

The wsRoleList element returns a list of roles available for the user specified in the passport. You can then use this list of roles to execute different routines depending on available roles or to re-login with a different role.

Element Name	XSD Type	Notes
userID	RecordRef	References an existing user record.
role	RecordRef	References an existing role record.
isDefault	boolean	If true, the role is set as the default role for this user.
isInactive	boolean	If true, this role is currently unavailable for the current user.
isLoggedInRole	boolean	If true, the role of the logged in user is used.

Retrieving userID in Axis 1.1

Following is sample code that illustrates how you can retrieve the userID from the header when using Axis 1.1 — where **port** is the service port for a particular endpoint.

```
String userid = getHeader(port, "sessionInfo");
public static String getHeader(NetSuitePortType port, String headerName) {
   com.netsuite.webservices.platform.NetSuiteBindingStub stub =
   (com.netsuite.webservices.platform.NetSuiteBindingStub) port;
   SOAPHeaderElement [] headers = stub.getResponseHeaders();
   for (int i=0; i< headers.length; i++)
   {
      SOAPHeaderElement header = headers[i];
      if (header.getName().equals(headerName))
      {
         Iterator childElements = header.getChildElements();
         while (childElements.hasNext())
         {
            SOAPElement el = (SOAPElement) childElements.next();
            return el.getValue();
         }
      }
   }
   return null;
}
```

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InsufficientPermissionFault
- InvalidAccountFault
- InvalidSessionFault
- InvalidCredentialsFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In the following example a user is logged in with the Administrator role as indicated by the internalId of 3. For a list of available role IDs see Internal IDs Associated with Roles.

```
<soap:Body>
<platformMsgs:login>
   <platformMsgs:passport>
      <platformCore:email>jsmith@yahoo.com</platformCore:email>
      <platformCore:password>password123</platformCore:password>
      <platformCore:account>121511</platformCore:account>
      <platformCore:role internalId="3"/>
</platformMsgs:passport>
```

```
</platformMsgs:login>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
<loginResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<sessionResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"/>

   <ns2:wsRoleList xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">
      <ns2:wsRole>
         <ns2:role internalId="3">
            <ns2:name>Administrator</ns2:name>
         </ns2:role>
         <ns2:isDefault>false</ns2:isDefault>
         <ns2:isInactive>false</ns2:isInactive>
      </ns2:wsRole>
      <ns2:wsRole>
         <ns2:role internalId="15">
            <ns2:name>Employee Center</ns2:name>
         </ns2:role>
         <ns2:isDefault>false</ns2:isDefault>
         <ns2:isInactive>false</ns2:isInactive>
      </ns2:wsRole>
      <ns2:wsRole>
         <ns2:role internalId="22">
            <ns2:name>Intranet Manager</ns2:name>
         </ns2:role>
         <ns2:isDefault>false</ns2:isDefault>
         <ns2:isInactive>false</ns2:isInactive>
      </ns2:wsRole>
      <ns2:wsRole>
         <ns2:role internalId="25">
            <ns2:name>System Administrator</ns2:name>
         </ns2:role>
         <ns2:isDefault>false</ns2:isDefault>
         <ns2:isInactive>false</ns2:isInactive>
      </ns2:wsRole>
   </ns2:wsRoleList>
</sessionResponse>
</loginResponse>
</soapenv:Body>
```

C#

```
private void login( bool isAuto )
{
if ( !_isAuthenticated )
   {
   // Check whether this is a forced login as part of another operation
      if ( isAuto )
      _out.writeLn( "\nYou need to first login before invoking this operation ..." );
      // Enable client cookie management. This is required.
      _service.CookieContainer = new CookieContainer();

      // Populate Passport object with all login information
      Passport passport = new Passport();
      RecordRef role = new RecordRef();

      // Determine whether to get login information from config
      // file or prompt for it
      if ( "true".Equals( _dataCollection["promptForLogin"] ) )
      {
         _out.writeLn( "\nPlease enter your login information: " );
         _out.write( "E-mail: " );
         passport.email = _out.readLn();
         _out.write( "Password: " );
         passport.password = _out.readLn();
         _out.write( "Role nsKey (press enter for default administrator role): " );
         role.internalId = _out.readLn();
         passport.role =role;
         _out.write( "Account: " );
         passport.account = _out.readLn();
      }
      else
      {
         passport.email = _dataCollection["login.email"];
         passport.password = _dataCollection["login.password"];
         role.internalId = _dataCollection["login.roleNSkey"];
         passport.role =role;
         passport.account= _dataCollection["login.acct"];
      }

      // Login to NetSuite
      _out.info( "\nLogging into NetSuite" );
      _out.info( "Username: " + passport.email );
      _out.info( "Account: " + passport.account );
      Status status = (_service.login( passport )).status;

      // Process response
      if ( status.isSuccess == true )
      {
         _isAuthenticated = true;
         _out.info( "\nThe user with nsKey=" + _service.sessionInfo.userId + " was logged
         in successful and a new session has been created." );
      }
      else
      {
         // Should never get here since any problems with the
         // login should have resulted in a SOAP fault
         _out.error( getStatusDetails( status ) );
      }
   }
}
```

Java

```
public void login(boolean isAuto) throws RemoteException {
if (!_isAuthenticated) {
   // Check whether this is a forced login as part of another operation
   if (isAuto)
      _console
      .writeLn("\nYou need to first login before invoking this operation ...");

      // Populate Passport object with all login information
      Passport passport = new Passport();
      RecordRef role = new RecordRef();

      // Determine whether to get login information from config
      // file or prompt for it
      if ("true".equals(_properties.getProperty("promptForLogin"))) {
         _console.writeLn("\nPlease enter your login information: ");
         System.out.print("E-mail: ");
         passport.setEmail(_console.readLn());
         System.out.print("Password: ");
         passport.setPassword(_console.readLn());
         System.out.print("Role nsKey (press enter for default administrator role): ");
         role.setInternalId(_console.readLn());
         passport.setRole(role);
         System.out.print("Account: ");
         passport.setAccount(_console.readLn());
      } else {
         passport.setEmail(_properties.getProperty("login.email"));
         passport.setPassword(_properties.getProperty("login.password"));
         role.setInternalId(_properties.getProperty("login.roleNSkey"));
         passport.setRole(role);
         passport.setAccount(_properties.getProperty("login.acct"));
      }

      // Login to NetSuite
      _console.info("\nLogging into NetSuite");
      _console.info("Username: " + passport.getEmail());
      _console.info("Account: " + passport.getAccount());

      Status status = (_port.login(passport)).getStatus();
      // Process the response
      if (status.isIsSuccess() == true) {
         _isAuthenticated = true;
         _console
         .info("\nThe login was successful and a new session has been created.");
      } else {
         // Should never get here since any problems with the
         // login should have resulted in a SOAP fault
         _console.error(getStatusDetails(status));
      }
   }
}
```

logout

The logout operation is used to terminate an active session.

Note: If you explicitly log out of a session, and then attempt to utilize the same session, the SESSION_TIMED_OUT error message is returned.

Request

The logoutRequest type is used for the request. It does not contain any fields.

Response

The status type is used for the response. It does not contain any fields.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidCredentialsFault
- InvalidSessionFault
- ExceededRequestLimitFault
- UnexpectedErrorFault

Sample Code

Following is an example that contains an excerpt of the SOAP body for both the request and response.

SOAP Request

```
<logout/>
```

SOAP Response

```
<logoutResponse>
   <status isSuccess=”true”/>
</logoutResponse>
```

C#

```
private void logout()
{
   if ( _isAuthenticated )
   {
      _out.info( "\nLogging out of NetSuite\n" );
```

```
        // Logout from NetSuite
        Status status = (_service.logout()).status;

        if ( status.isSuccess == true )
        {
            _isAuthenticated = false;
            _out.info( "Logout successful" );
        }
        else
        {
            // Should never get here since any problems with the
            // logout should have resulted in a SOAP fault
            _out.error( getStatusDetails( status ) );
        }
    }
    else
    {
        _out.info(
        "\nThe logout() operation cannot be invoked because there is no active session. " +
        "You must be first logged on before attempting to logout.\n" );
    }
}
```

Java

```
public void logout() throws RemoteException {
    if (_isAuthenticated) {
        _console.info("\nLogging out of NetSuite\n");

        // Logout from NetSuite
        Status status = (_port.logout()).getStatus();

        if (status.isIsSuccess() == true) {
            _isAuthenticated = false;
            _console.info("Logout successful");
        } else {
            // Should never get here since any problems with the
            // logout should have resulted in a SOAP fault
            _console.error(getStatusDetails(status));
        }
    } else {
        _console
        .info("\nThe logout() operation cannot be invoked because there is no active session. "
        + "You must be first logged on before attempting to logout.\n");
    }
}
```

mapSso

Single sign-on (SSO) refers to the procedure that allows a user of two or more user-authenticating Web applications to move between these applications using a browser, only presenting **authentication** information one time per session.

The mapSso operation supports the inbound single sign-on feature. This feature allows users to go directly from an external user-authenticating application to NetSuite, without having to separately log in to NetSuite. Validation is provided by the passing of an encrypted token, and user identification is provided by a mapping between external application credentials (remote company ID and remote user ID), and NetSuite credentials (email, password, account, and role used to log in to NetSuite).

The mapSso operation streamlines the provisioning of inbound single sign-on access to NetSuite accounts by automating the mapping of credentials between an external application and NetSuite. If this operation is not used, this mapping is created for each user by requiring them to log in to NetSuite on their first single sign-on access.

This operation allows users to access NetSuite through inbound single sign-on without knowing their NetSuite password. Use of this operation is required for inbound single sign-integrations to Web store.

Important: This operation provides mapping, not a login to NetSuite. This operation does NOT provide the ability to log in using a single sign-on token, cannot provisian a partner with the inbound single sign-on feature, or aid with public/private key exchange. Use of this operation implies that the account already has inbound single sign-on set up and has access to the associated partner ID and encryption keys needed to generate the token.

Be aware of the following:

- If a NetSuite role used for inbound single sign-on is deleted, the single sign-on mapping for any user with that role is automatically remapped to another role.
- If a user has a single sign-on mapping set up with a particular role and that role is removed from the user, the mapping is deleted. You can set up a new mapping for that user with a different role.
- Single sign-on mappings are not copied from a production account to a sandbox account when the sandbox is refreshed. These mappings must be recreated in the sandbox account for any users who require inbound single sign-on access to that account.

For more detailed information on the inbound single sign-on feature, see the help topic Inbound Single Sign-on.

Important: NetSuite hosts customer accounts in multiple data centers. For that reason, the correct URL for web services access varies depending on the data center hosting the account. Your integration **must** incorporate logic that dynamically determines the correct URL. With the 2012.2 and later endpoints, you should use the getDataCenterUrls operation to dynamically discover the correct URL. With older endpoints, you should use the REST roles service. For details, see the help topic Using the REST roles Service to Get User Accounts, Roles, and Domains.

Request

The MapSsoRequest type is used for this request. It contains the following fields:

Element Name	XSD Type	Notes
ssoCredentials	SsoCredentials	Contains all the required credentials including username (email address), password, account, and role, to authenticate the user and create a new session.

The following table describes the fields in the SsoCredentials complex type. All are required.

Field	Notes
email	This string is the email address used to log in to NetSuite.
password	This string is the password used to log in to NetSuite.
account	This string is the NetSuite account ID. **Note:** You can confirm your account number in the NetSuite UI. As administrator, go to Support > Customer Service > Contact Support by Phone. Your account number is displayed in a pop-up box.
role	This references an existing role for the user. (The XSD type is RecordRef.)
authentication token	This is a string representing the encrypted token. This token, prior to any hex-encoding, is of the form: `<companyID><space><userID><space><timestamp>` **Note:** As spaces are used to delimit subtokens, the values for companyID and userID cannot include spaces. For more information about the values in the authentication token, see the Inbound Single Sign-on documentation, in particular: Parameters used for both Application and Web Store Access.
partner ID	This is the affiliate ID of the integration partner, as provided to the partner by NetSuite.

Response

The MapSsoResponse type is used for the response. It does not contain any fields.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidAccountFault
- InvalidCredentialsFault
- InvalidVersionFault
- ExceededRequestLimitFault
- UnexpectedErrorFault

Sample Code

SOAP Request

```
<soapenv:Envelope
    xmlns:xsd='http://www.w3.org/2001/XMLSchema'
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/'
    xmlns:platformCore='urn:core_2017_1.platform.webservices.netsuite.com'
    xmlns:platformMsgs='urn:messages_2017_1.platform.webservices.netsuite.com'>
    <soapenv:Header/>
    <soapenv:Body>
        <mapSso xsi:type='platformMsgs:MapSsoRequest'>
            <ssoCredentials xsi:type='platformCore:SsoCredentials'>
                <email xsi:type='xsd:string'>user@example.com</email>
                <password xsi:type='xsd:string'>ExamplePassword1</password>
                <account xsi:type='xsd:string'>TSTDRV123456</account>
                <role xsi:type='platformCore:RecordRef' internalId='3'/>
                <authenticationToken xsi:type='xsd:string'>201F20C54BB7B134081CA1F04D3A9EC0CC4F
0C6C74FEDE16F18BE0AF5A41CF7372949C777574266364187AF1BD3BFA31D105F83D39BE69EC9525475D95F0B3C9830
3DA2985AA6DBF28F0F027D88461A3E68F0E414CCC27CD2C0B127B8192D81D17FF7555D1D84B42832BD9ABC9556FCA6E
E1E19E0EA5AABDFC4BE599B7D798CF</authenticationToken>
                <partnerId xsi:type='xsd:string'>123456</partnerId>
            </ssoCredentials>
        </mapSso>
    </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://w
ww.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Header>
        <platformMsgs:documentInfo xmlns:platformMsgs="urn:messages_2017_1.platform.webservices
.netsuite.com">
            <platformMsgs:nsId>WEBSERVICES_TSTDRV123456_111420161694237117594769456_3bf28e</pla
tformMsgs:nsId>
        </platformMsgs:documentInfo>
    </soapenv:Header>
    <soapenv:Body>
        <mapSsoResponse xmlns="">
            <platformMsgs:sessionResponse xmlns:platformMsgs="urn:messages_2017_1.platform.webs
ervices.netsuite.com">
                <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platf
orm.webservices.netsuite.com" />
                <platformMsgs:userId internalId="-5" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com">
                    <platformCore:name>TheAdministrator</platformCore:name>
                </platformMsgs:userId>
                <platformCore:wsRoleList xmlns:platformCore="urn:core_2017_1.platform.webservic
es.netsuite.com">
                    <platformCore:wsRole>
                        <platformCore:role internalId="3">
                            <platformCore:name>Administrator</platformCore:name>
                        </platformCore:role>
                        <platformCore:isDefault>false</platformCore:isDefault>
                        <platformCore:isInactive>false</platformCore:isInactive>
                        <platformCore:isLoggedInRole>true</platformCore:isLoggedInRole>
                    </platformCore:wsRole>
                </platformCore:wsRoleList>
            </platformMsgs:sessionResponse>
        </mapSsoResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Java

```
 /* Generate a NetSuitePort */
   NetSuiteServiceLocator nss = new NetSuiteServiceLocator();
   NetSutePortType myNetSuitePort = nss.getNetSuitePort();

/* Here your code needs to generate a valid Netsuite Sso token */
   String mySsoToken = MySSOManager.getTokenFromIds
   ("myCompanyID","thisUsersUIDinMySystem");
   RecordRef rr = new RecordRef();
   rr.setid("23");

 /* Setup the Credential */
   SsoCredentials sc = new SsoCredentials();
   sc.setAccount("TSTDRV00000");
   sc.setEmail("bob@happycow.com");
   sc.setPassword("fr0mCA");
   sc.setRole(rr);
   sc.setAuthenticationToken(mySsoToken);

 /* Now initiate the mapping*/
   try{
      SessionResponse sr = myNetSuitePort.mapSso(sc);
      if (!sr.getStatus().isIsSuccess())
      throw new Exception("Mapping Failed: " + sr.getStatus().getStatusDetail(0).getMessage());

}
```

search

The *search* operation is used to execute a search on a specific record type based on a set of criteria. You can search by defining search filter fields on the record, joined fields on an associated record, search return columns, or joined search return columns from an associated record. The results of the search can be complete records, or a select set of fields specified through search return columns.

Note that you can also use the *search* operation to return an existing saved search. You cannot use the search operation to retrieve state values. You must use the getAll operation to retrieve all state values in the system. The getAll operation will return **all** states, not the legal ones for your default country. Also note that the country and state must match on the address.

Important: Be aware that the search preferences you set in SearchPreferences object affect the search request and response. See Web Services Preferences for details.

Use the *search* operation to execute the following types of searches:

- **Basic search** — Execute a search on a record type based on search filter fields that are specific to that type. See Basic Searches in Web Services.
- **Joined search** — Execute a search on a record type based on search filter fields on an associated record type. See Joined Searches in Web Services

- **Advanced search** — Execute a search on a record type in which you specify search filter fields **and/or** search return columns or joined search columns. Using advanced search, you can also return an existing saved search. See Advanced Searches in Web Services.

Note: An asynchronous equivalent is available for the search operation, **asyncSearch**. For information about asynchronous request processing, see Synchronous Versus Asynchronous Request Processing.

Also see the following sections for information on setting additional search filtering values and searching for custom fields:

- Setting Valid Search Values
- Setting the anyof, mine, or myteam Filtering Values
- Searching by lastModifiedDate
- Understanding Sorting in Advanced Search
- Search-Related Sample Code
- Searching for a Multi-select Custom Field

Important: To go directly to search-related code samples, see Search-Related Sample Code.

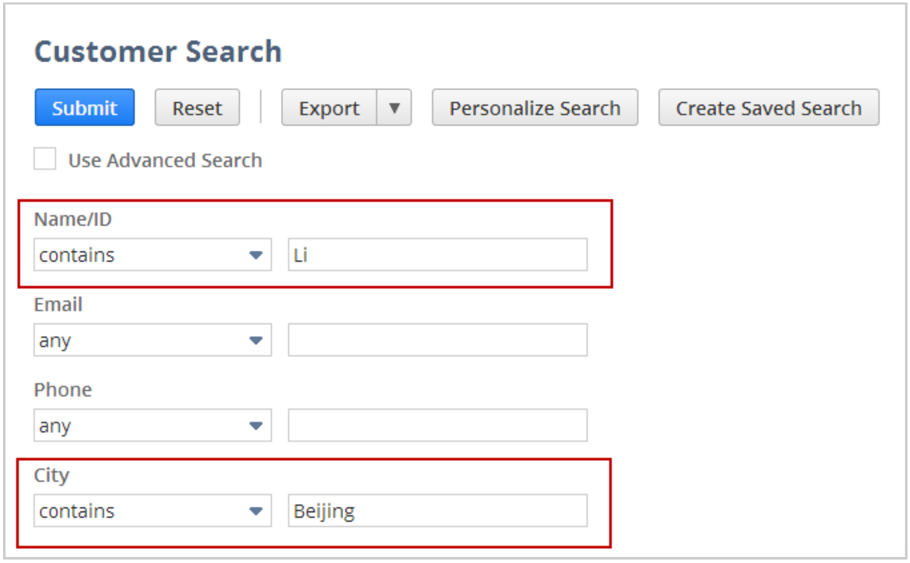
Basic Searches in Web Services

See the following topics to learn how to execute a basic search in NetSuite using web services:

- What is a basic search?
- Which SuiteTalk objects are used in a basic search?
- Basic Search Code Sample

What is a basic search?

A basic search lets you search records of a specific type using the fields on that record as search filters. The following figure shows the UI equivalent of a basic Customer search. Start by going to Lists > Relationships > Customers > Search, and ensure that the Use Advanced Search box is not checked. You can specify one or more field values to use as filters for search results.


Customer Search

Use Advanced Search

Name/ID: contains — Li

Email: any

Phone: any

City: contains — Beijing

In a basic search, field criteria are the **only** values you set. You cannot specify search return columns . In web services, specifying search return columns is the equivalent of performing an advanced search. (See Advanced Searches in Web Services for details.)

In the example of a basic Customer search (see previous figure), the results returned include the record ID of every customer that has the Category field set to *From Advertisement* and the Status field set to *Customer-Closed Won*. Note that **ALL** the other data associated with these specific customer records are returned as well. As a consequence, in web services a basic search tends to increase the search response time.

Which SuiteTalk objects are used in a basic search?

To perform a basic search in which you specify search filter criteria only, use:

1. < *Record* > **Search**
2. < *Record* > **SearchBasic**

For more details, see Basic Search Objects Explained and Basic Search Code Sample.

Basic Search Objects Explained

In SuiteTalk, any record that supports search has a corresponding < *Record* > **Search** object, which contains a *basic* element. The *basic* element references a < *Record* > **SearchBasic** object, which defines all available search criteria (filter fields) specific to that record type.

The XSD snippet below shows the CustomerSearch object. The *basic* element references the CustomerSearchBasic object, which defines all available search criteria for the Customer record.

```
<complexType name=" CustomerSearch ">
        <complexContent>
            <extension base="platformCore:SearchRecord">
                <sequence>
                    <element name=" basic " type="platformCommon: CustomerSearchBasic " minOccu
rs="0"/>
                    <element name="callJoin" type="platformCommon:PhoneCallSearchBasic" minOccu
rs="0"/>
                    <element name="campaignResponseJoin" type="platformCommon:CampaignSearchBas
ic"
         minOccurs="0"/>
                 .....

      </sequence>
            </extension>
        </complexContent>
    </complexType>
```

Note: < *Record* > **Search** objects reside in the same XSD as their corresponding record objects. In this case, both the Customer and CustomerSearch objects reside in the listRel XSD. Also note that the CustomerSearch object, like all < *Record* > **Search** objects, provide available search joins for that record type. For information on joined searches, see Joined Searches in Web Services.

This snippet shows the CustomerSearchBasic object. All < *Record* > **SearchBasic** objects are defined in the platformCommon XSD. This sample shows four of the available fields (accountNumber, address, addressee, addressLabel) that can be used as search criteria on the Customer record.

```
<complexType name=" CustomerSearchBasic ">
      <complexContent>
   <extension base="platformCore:SearchRecord">
      <sequence>
        <element name=" accountNumber " type="platformCore:SearchStringField" minOccurs="0" />

        <element name=" address " type="platformCore:SearchStringField" minOccurs="0" />
       <element name=" addressee " type="platformCore:SearchStringField" minOccurs="0" />
        <element name=" addressLabel " type="platformCore:SearchStringField" minOccurs="0" />
               ...
     </sequence>
         </extension>
   </complexContent>
</complexType>
```

For a code sample of a basic search, see Basic Search Code Sample.

Joined Searches in Web Services

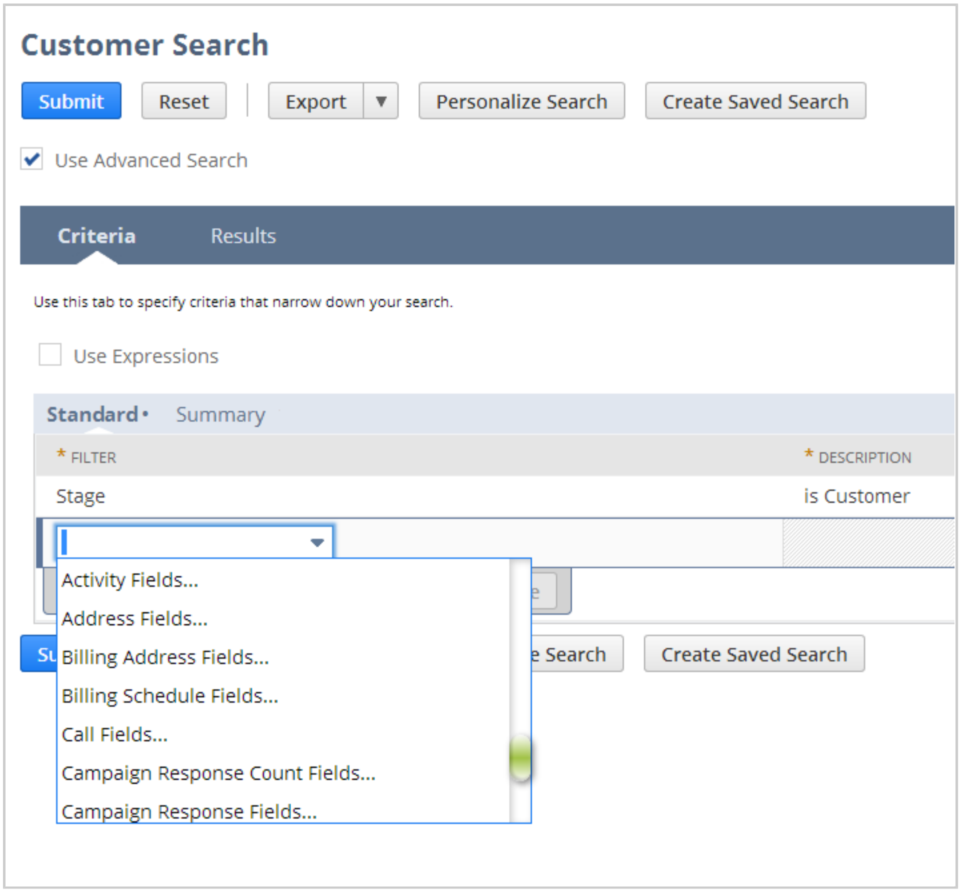
See the following topics to learn how to execute a joined search in NetSuite using web services:

- What is a joined search?
- Which SuiteTalk objects are used in a joined search?
- Joined Search Code Samples

What is a joined search?

A joined search allows you search against a specific record type using the fields on an associated record as search filters. In the UI, you can identify which associated records provide joined filter criteria by first navigating to a record's search interface. For example, for the Customer search interface, go to Lists > Relationships > Customers > Search.

Check the Use Advanced Search box and wait for the page to update. Then scroll through the Filter dropdown list. Joined search records are indicated by the record name followed by an ellipsis (....). Search fields from any of the records listed (that are also currently exposed in SuiteTalk) can be included in the search criteria.



Note: For a list of SuiteTalk-supported records, see the help topic Web Services Supported Records.

Which SuiteTalk objects are used in a joined search?

To perform a joined search in which you use search filter criteria from an associated record, use:

1. < *Record* > **Search**
2. < *Record* > **SearchBasic**

For more details, see Joined Search Objects Explained and Joined Search Code Samples.

Joined Search Objects Explained

In SuiteTalk, all search joins are listed in the < *Record* > **Search** object. For example, to find available search joins for the Contact or Employee record, see the ContactSearch and EmployeeSearch XSDs, respectively.

The snippet below shows the CustomerSearch object, which includes < *xxx* > **Join** elements. These elements reference search criteria available from other SuiteTalk-supported records.

```
<complexType name=" CustomerSearch ">
        <complexContent>
            <extension base="platformCore:SearchRecord">
```

```
                <sequence>
                    <element name="basic" type="platformCommon:CustomerSearchBasic" minOccurs="
0"/>
                    <element name=" callJoin " type="platformCommon: PhoneCallSearchBasic " min
Occurs="0"/>
                    <element name=" campaignResponseJoin " type="platformCommon: CampaignSearch
Basic "
         minOccurs="0"/>
                    <element name=" caseJoin " type="platformCommon: SupportCaseSearchBasic " m
inOccurs="0"/>
      .....

       </sequence>
             </extension>
         </complexContent>
     </complexType>
```

In this case, all search filter criteria from the PhoneCallSearchBasic, CampaignSearchBasic, and SupportCaseSearchBasic objects are available to the Customer record as joined search filters. Note that all < *Record* > **SearchBasic** objects in NetSuite web services are defined in the platformCommon XSD.

For a code sample of a joined search, see Joined Search Code Samples.

Important: Only fields on **SuiteTalk-supported** records can be specified as filter criteria for a joined search request. For a list of SuiteTalk-supported records, see the help topic Web Services Supported Records.

Returning an Associated Joined List of Records

Using a combination of joined search and the internalId list on each record, you can retrieve a list of records for an associated list of records. For example, you can retrieve a list of contacts for a specific list of customers. To do this, you must first retrieve the desired list of internalIds for the record you need to retrieve by, and then submit that list in a joined search query to retrieve the associated list.

See Joined Search Code Samples for an example of a joined search which uses a list of records.

Advanced Searches in Web Services

Advanced searching in SuiteTalk provides users with the ability to:

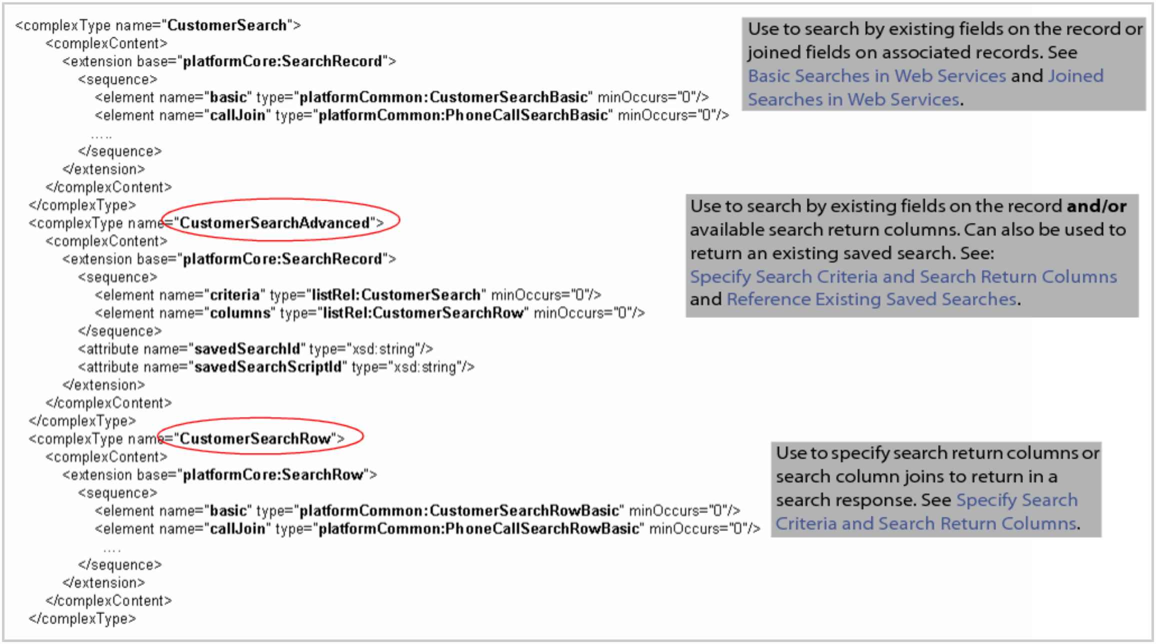
- Perform a search that references an existing saved search. See Reference Existing Saved Searches.
- Perform a search that references an existing saved search, and then overrides existing search return columns with new search return columns. See Specify Search Criteria and Search Return Columns.
- Perform a search that references an existing saved search, and then provides additional search filter criteria (on top of the criteria already specified in the saved search). See Specify Search Criteria and Search Return Columns.
- Perform a search that specifies search criteria and search result columns. See Specify Search Criteria and Search Return Columns.

The SuiteTalk API includes advanced search objects for all records that have an existing search interface. To see which objects are used to execute advanced searches, see Which SuiteTalk objects are used in advanced search?

For advanced search code samples, see Advanced Search Code Samples.

Important: Note that advanced search functionality will not work in endpoints prior to 2008.2. Also note that searchMore, searchNext, and respective asynchronous operations can be used in advanced search, as well as all search preferences and pagination functionality.

The following is an XSD snippet of the listRel XSD. This sample provides a high-level overview of advanced search objects and their relationship to the basic search object < *Record* > **Search**. In this sample, the objects Customer **SearchAdvanced**, Customer **SearchRow**, and Customer **SearchRowBasic** (not shown in sample) are considered to be advanced search objects in the SuiteTalk API.



```
<complexType name="CustomerSearch">
    <complexContent>
      <extension base="platformCore:SearchRecord">
        <sequence>
          <element name="basic" type="platformCommon:CustomerSearchBasic" minOccurs="0"/>
          <element name="callJoin" type="platformCommon:PhoneCallSearchBasic" minOccurs="0"/>
          .....
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="CustomerSearchAdvanced">
    <complexContent>
      <extension base="platformCore:SearchRecord">
        <sequence>
          <element name="criteria" type="listRel:CustomerSearch" minOccurs="0"/>
          <element name="columns" type="listRel:CustomerSearchRow" minOccurs="0"/>
        </sequence>
        <attribute name="savedSearchId" type="xsd:string"/>
        <attribute name="savedSearchScriptId" type="xsd:string"/>
      </extension>
    </complexContent>
  </complexType>
  <complexType name="CustomerSearchRow">
    <complexContent>
      <extension base="platformCore:SearchRow">
        <sequence>
          <element name="basic" type="platformCommon:CustomerSearchRowBasic" minOccurs="0"/>
          <element name="callJoin" type="platformCommon:PhoneCallSearchRowBasic" minOccurs="0"/>
          ....
        </sequence>
      </extension>
    </complexContent>
  </complexType>
```

Use to search by existing fields on the record or joined fields on associated records. See Basic Searches in Web Services and Joined Searches in Web Services.

Use to search by existing fields on the record **and/or** available search return columns. Can also be used to return an existing saved search. See: Specify Search Criteria and Search Return Columns and Reference Existing Saved Searches.

Use to specify search return columns or search column joins to return in a search response. See Specify Search Criteria and Search Return Columns.

Note: For additional SOAP and code samples related to searches, see Search-Related Sample Code, Joining Through Custom Fields, and Searches.

Reference Existing Saved Searches

Advanced search in web services lets you reference an existing saved search. Returning saved searches provides you with access to data that otherwise could not be returned in SuiteTalk. For example, advanced search lets you return saved searches that include formula filters (in the search criteria) or expressions.

When working with saved searches, two questions come to mind for web services users:

- Why reference an existing saved search?
- How do I reference an existing saved search?

Why reference an existing saved search?

The following use cases illustrate possible scenarios for referencing a saved search:

- You want to return only a subset of data from a record (in other words, data that is specified through the saved search's return columns).

- Your integration application processes a set of records that are identified in a saved search. Periodically, users change the criteria of the search. By referencing a saved search ID, your code can reference their saved search. Developers do not have to change and re-test code every time the search criteria changes.
- You have a complex search that compiles data from many different records. You can create a saved search in the NetSuite UI, and then reference this search in web services rather than try to code the search in web services.
- You want to reference an existing saved search based on Leads, for example. You can return all the data provided in this search, and then define additional criteria for the search response. For example, you can return a Leads saved search and then provide additional criteria that returns the leads from this search created with today's date. In other words, if you reference a saved search and add any filter criteria to the search request, the additional criteria will be conjunctive with the saved search criteria.

How do I reference an existing saved search?

First you must obtain the saved search ID. You can do so through the UI by going to Lists > Search > Saved Searches. The saved search ID appears in the ID column.

You can also use the getSavedSearch operation to programmatically retrieve a list of saved search IDs for a specific record type. Note that this operation does nothing more than return a list of saved search IDs on a per-record-type basis (for example, all saved search IDs for the customer record type).

You can then use the search() operation, along with the < *Record* > **SearchAdvanced** object to return the details of the saved search. The following is a example that shows how to instantiate the CustomerSearchAdvanced object and specify a savedSearchId to return.

```
// create search object
CustomerSearchAdvanced customerSearch = new CustomerSearchAdvanced();

//set saved search id
customerSearch.savedSearchId="100";

// perform the search
NetSuiteService nss = new NetSuiteService();
SearchResult result = nss.search(customerSearch);
```

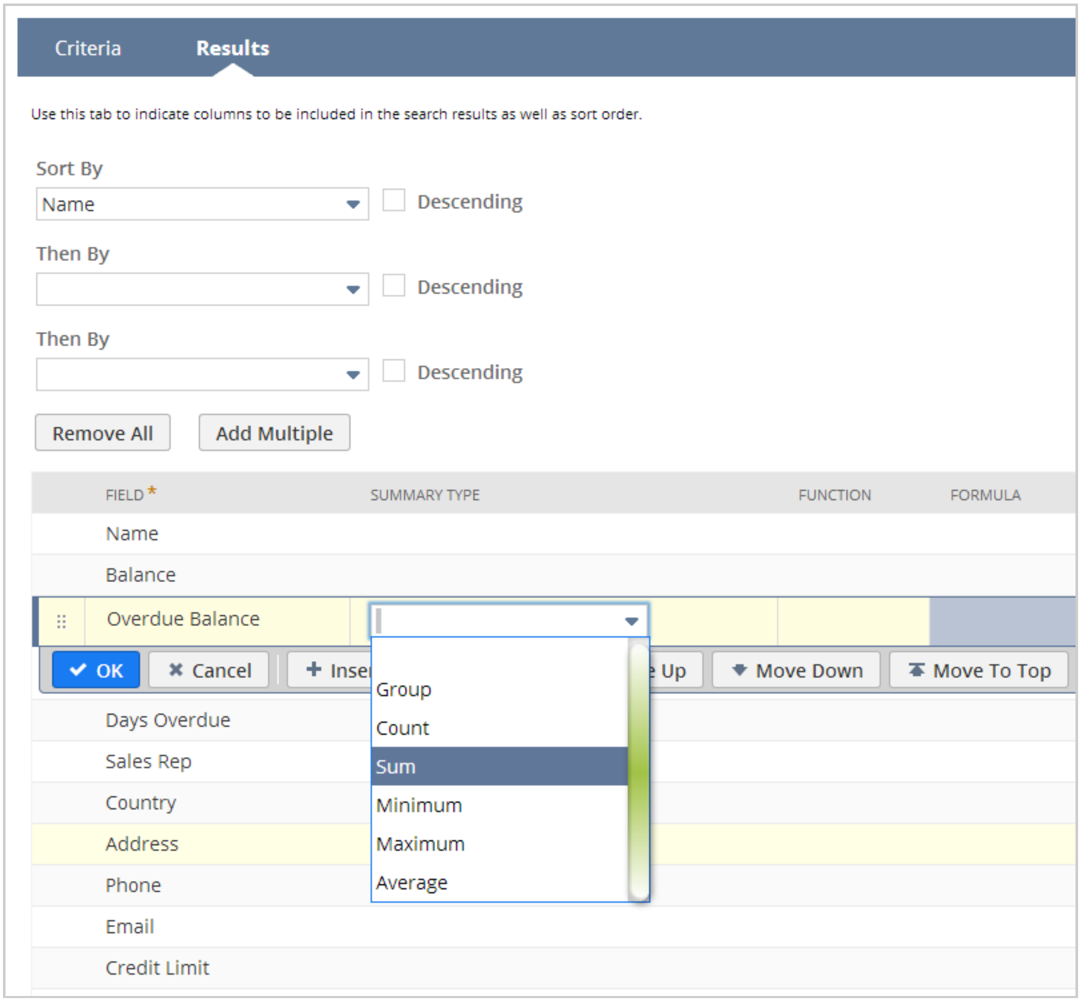
For more detailed samples, see Advanced Search Code Samples.

Usage Notes

- Only one saved search can be referenced as part of the search call.
- If you reference a saved search that contains search functions, you *will* get the results back. However, you cannot *create* an advanced search that uses search functions. Creating a search in web services that uses search functions is not currently supported.
- If you reference a saved search for which you do not have permission, the exception is ignored and fields have no value (are not in the SOAP response at all).
- If you reference a saved search that contains summary results, you will get the following error:

  We cannot return search columns for summary saved search < *saved search ID* >

  The following figure shows the UI equivalent of setting a summary type on a search return column. If a value is set, the saved search cannot be returned.



Use this tab to indicate columns to be included in the search results as well as sort order.

FIELD*	SUMMARY TYPE	FUNCTION	FORMULA
Name			
Balance			
Overdue Balance			
Days Overdue			
Sales Rep			
Country			
Address			
Phone			
Email			
Credit Limit			

Specify Search Criteria and Search Return Columns

Similar to a basic search, an advanced search allows you search against a specific record type using the fields on that record as search filters. In addition, you can **also** specify a set of search return columns or joined columns to return.

This kind of advanced search is useful for retrieving only the record data you need rather than the contents of an entire record.

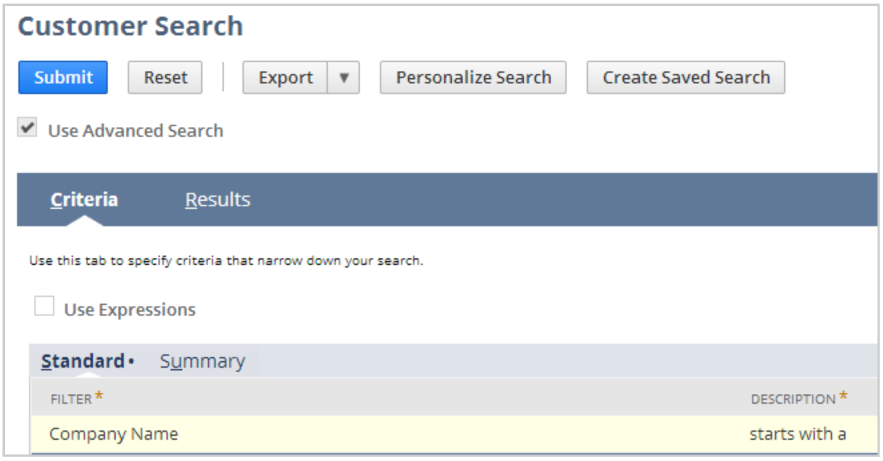
Advanced searches work well, for example, for users who may have a mobile client that needs to display only the name, phone number, and email address of their sales rep contact. Rather than returning all of the data on a contact record, users can create an advanced search that pulls only the relevant information. They are no longer required to download the entire record.

For advanced search code samples that show how to specify both search criteria and search return columns, see Advanced Search Code Samples.

What are search return columns?

The following figures show the UI equivalent of an advanced search that includes search return columns.

1. First, specify the basic search criteria on the Criteria subtab (in this example all customers that have a company name starting with the letter A).



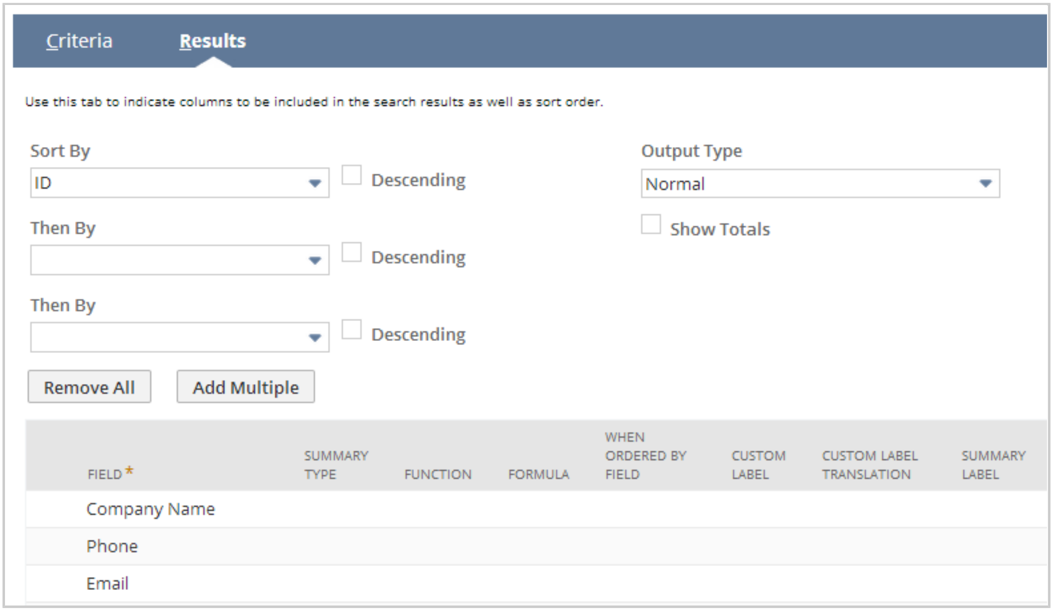
☑ Use Advanced Search

Use this tab to specify criteria that narrow down your search.

☐ Use Expressions

FILTER *	DESCRIPTION *
Company Name	starts with a

2. Next, click the Results subtab to define the search return columns. When your search is executed, your search response will return only the company name, phone, and contact name of all customers whose company starts with the letter A.

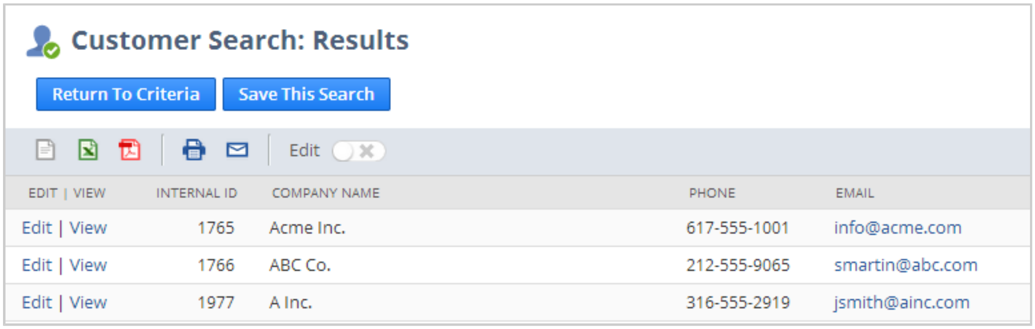


Use this tab to indicate columns to be included in the search results as well as sort order.

Sort By: ID ☐ Descending
Then By: ☐ Descending
Then By: ☐ Descending
Output Type: Normal
☐ Show Totals

FIELD *	SUMMARY TYPE	FUNCTION	FORMULA	WHEN ORDERED BY FIELD	CUSTOM LABEL	CUSTOM LABEL TRANSLATION	SUMMARY LABEL
Company Name							
Phone							
Email							

The following figure shows the results of the search. Only the relevant data are returned.



EDIT | VIEW	INTERNAL ID	COMPANY NAME	PHONE	EMAIL
Edit | View	1765	Acme Inc.	617-555-1001	info@acme.com
Edit | View	1766	ABC Co.	212-555-9065	smartin@abc.com
Edit | View	1977	A Inc.	316-555-2919	jsmith@ainc.com

Which SuiteTalk objects are used in advanced search?

The following summarizes the objects used in different types of advanced searches. For additional details, see Advanced Search Objects Explained and Advanced Search Code Samples.

- To perform a search that specifies search filter criteria and search result columns, use:
    1. < *Record* > **Search**
    2. < *Record* > **SearchBasic**
    3. < *Record* > **SearchRow**
    4. < *Record* > **SearchRowBasic**
- To perform a search that references an existing saved search, use:
    1. < *Record* > **SearchAdvanced**
- To perform a search that references an existing saved search, and then overrides existing search return columns with new search return columns, use:
    1. < *Record* > **SearchAdvanced**
    2. < *Record* > **SearchRow**
    3. < *Record* > **SearchRowBasic**
- To perform a search that references an existing saved search, and then provides additional search filter criteria (that is in *addition* to the criteria already specified in the saved search), use:
    1. < *Record* > **SearchAdvanced**
    2. < *Record* > **Search**
    3. < *Record* > **SearchBasic**

Advanced Search Objects Explained

The < Record >SearchAdvanced object contains:

- a *criteria* element - references the < *Record* > **Search** object through which you specify standard search field criteria. (See Basic Searches in Web Services for details on the < *Record* >Search object.)
- *a columns* element - references the < *Record* > **SearchRow** object through which you specify a set of search result columns to return in the response.
- a *savedSearchId* attribute - references the saved search internal ID (for example, 57, 99, 63).

Note: If the Show Internal ID preference is enabled in your NetSuite account, the saved search internal ID appears in the **Internal ID** column in a saved search list. Programmatically, you can use getSavedSearch to obtain a list of saved search internal IDs for a specific record type.

- a *savedSearchScriptId* attribute - references a custom saved search ID (for example, customsearch_mySpecialSearch).

Note: If the Show Internal ID preference is enabled in your NetSuite account, the saved search ID appears in the **ID** column in a saved search list.

Important: You cannot use getSavedSearch to return a list of custom saved search IDs; only the system-defined internal IDs (57, 99, 63, etc.) will be returned.

The XSD sample below shows the CustomerSearchAdvanced object:

```
<complexType name=" CustomerSearchAdvanced ">
   <complexContent>
      <extension base="platformCore:SearchRecord">
         <sequence>
            <element name=" criteria " type="listRel:CustomerSearch" minOccurs="0"/>
            <element name=" columns " type="listRel:CustomerSearchRow" minOccurs="0"/>
            </sequence>
            <attribute name=" savedSearchId " type="xsd:string"/>
            <attribute name=" savedSearchScriptId " type="xsd:string"/>
         </extension>
```

The < Record >SearchRow object contains:

- a *basic* element - references the < *Record* > **SearchRowBasic** object, which specifies available search return columns and column joins for *that* record type.
- <xxx>Join elements - references the < *Record* > **SearchRowBasic** object, which specifies search return columns and column joins for the *associated* record type.

The XSD below shows a snippet of the CustomerSearchRow object:

```
<complexType name=" CustomerSearchRow ">
   <complexContent>
      <extension base="platformCore:SearchRow">
           <sequence>
             <element name=" basic " type="platformCommon:CustomerSearchRowBasic"
               minOccurs="0"/>
             <element name=" callJoin " type="platformCommon:PhoneCallSearchRowBasic"
            minOccurs="0"/>
             <element name=" campaignResponseJoin "type="platformCommon:CampaignSearchRowBasic"

            minOccurs="0"/>
         <element name=" caseJoin " type="platformCommon:SupportCaseSearchRowBasic"
            minOccurs="0"/>
            ......
          </sequence>
      </extension>
   </complexContent>
</complexType>
```

The < Record >SearchRowBasic object contains:

- search result column elements - use to define specific column names in your response.

The next snippet shows the CustomerSearchRowBasic object, which, like ALL < *Record* > **SearchRowBasic** objects, resides in the platformCommon XSD. This object lists all available search return columns for the Customer record.

```
<complexType name=" CustomerSearchRowBasic ">
        <sequence>
            <element name=" accountNumber " type="platformCore: SearchColumnStringField " minOc
curs="0"
      maxOccurs="unbounded"/>
```

```
            <element name=" address " type="platformCore: SearchColumnStringField " minOccurs="
0"
      maxOccurs="unbounded"/>
            <element name=" addressee " type="platformCore: SearchColumnStringField " minOccurs
="0"
      maxOccurs="unbounded"/>
            <element name=" addressLabel " type="platformCore: SearchColumnStringField " minOcc
urs="0"
      maxOccurs="unbounded"/>
           ...

        </sequence>
    </complexType>
```

Note that all search return columns reference SearchColumn< *xxx* >Field objects. See Search Column Custom XML Schema Types for definitions of each search column object.

Important: When executing an advanced search, you can set the SearchPreferences. *returnSearchColumns* preference to TRUE to ensure that only search return columns are returned in a search. An error is thrown if *returnSearchColumns* is set to TRUE and you have not specified search return columns in your request. (Note that you will **not** receive an error if you are using advanced search functionality to return a **saved search** that already includes search return columns.)

Note: Also note that in an advanced search, the *bodyFieldsOnly* preference is ignored.

The default value for *returnSearchColumns* is TRUE.

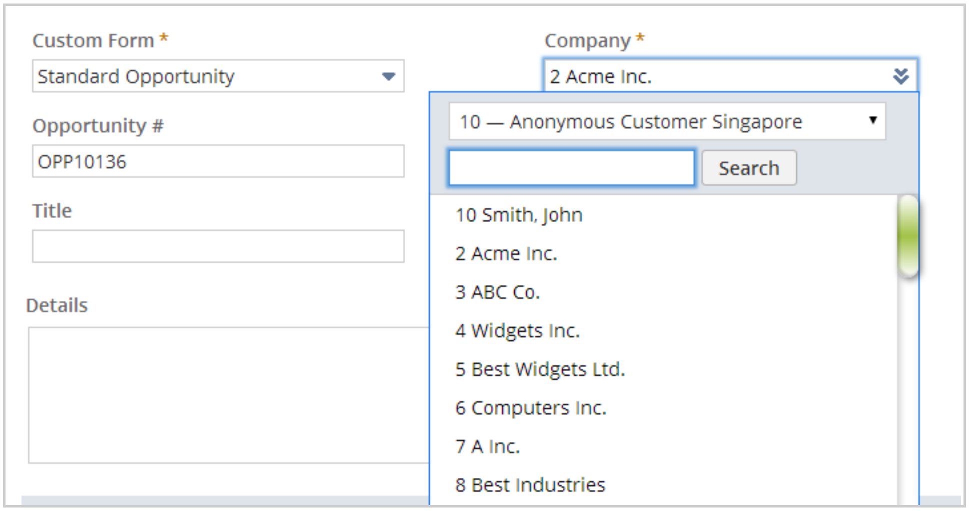
```
 <complexType name="SearchPreferences">
 <sequence>
  <element name="bodyFieldsOnly" minOccurs="0" type="xsd:boolean" default="true"/>
  <element name=" returnSearchColumns " minOccurs="0" type="xsd:boolean"
      default="true"/>
  <element name="pageSize" minOccurs="0" type="xsd:int"/>
 </sequence>
</complexType>
```

How Custom Fields Are Identified in Search Results

When executing an advanced search, you might choose to include as part of your search results values from custom fields. In some cases, the way that the system identifies these values varies depending on which endpoint you are using. For many users, this difference might not be of great consequence. However, you may want to review this information if you are upgrading from a previous endpoint to the 2013.2 WSDL or later, and if your integration depends on these values.

Specifically, this difference applies to scenarios in which you execute an advanced search, join to a record that has a custom field, and include the custom field in your search results. With 2013.1 and earlier endpoints, the system in its response adds a prefix to the string used to identify the custom field. In 2013.2 and later, this prefix is not added.

For example, suppose you had a custom item field on your inventory item record, and this custom field had an ID of custitem_style.



First, note that this value is referred to as internalId when using 2013.1 or an earlier endpoint. With 2013.2 and later endpoints, the value is called scriptId.

Second, with older endpoints, in some cases, the system modifies this value slightly. For example: Suppose you executed a TransactionSearchAdvanced and joined to the item record. If you included in your search results the custom field pictured above, the 2013.1 endpoint, or an earlier one, would identify the field's internalId as having a value of "item_custitem_style." That is, in this particular scenario, the system would add the "item_" prefix. With a different type of join, a different prefix would be added.

If you are using the 2013.2 endpoint or later, the system does **not** add a prefix. The scriptId returned in the response is the same as in the request ("custitem_style," in this example). Therefore, if you upgrade to 2013.2 or a later WSDL and if you have integrations that depend on this value, you must modify these integrations.

Note: For code samples in Java and C#, see Searching for a Multi-select Custom Field.

Example — Behavior with 2013.1 and Earlier Endpoints

In the following example, the user executes a TransactionSearchAdvanced operation with a join to the item record. The user is searching for estimates that are associated with inventory items. In the results, the user wants to include the value of the custom field with the internalId "custitem_color."

```
<soapenv:Body>
   <platformMsgs:search>
      <platformMsgs:searchRecord xsi:type="s0:TransactionSearchAdvanced">
         <s0:criteria>
            <s0:basic>
               <platformCommon:internalId operator="anyOf">
                  <platformCore:searchValue internalId="11962" type="estimate" />
               </platformCommon:internalId>
               <platformCommon:type operator="anyOf">
                  <platformCore:searchValue>_estimate</platformCore:searchValue>
               </platformCommon:type>
            </s0:basic>
            <s0:itemJoin>
               <platformCommon:type operator="anyOf">
```

```
                  <platformCore:searchValue>_inventoryItem</platformCore:searchValue>
               </platformCommon:type>
            </s0:itemJoin>
         </s0:criteria>
         <s0:columns>
            <s0:itemJoin>
               <platformCommon:customFieldList>
                  <platformCore:customField xsi:type="platformCore:SearchColumnSelectCustomFiel
d" internalId="custitem_color" />
              </platformCommon:customFieldList>
            </s0:itemJoin>
         </s0:columns>
      </platformMsgs:searchRecord>
   </platformMsgs:search>
</soapenv:Body>
```

The response might look like the following. Note that the internalId value is now “item_custitem_color.”

```
<platformCore:searchRowList>
   <platformCore:searchRow xsi:type="tranSales:TransactionSearchRow" xmlns:tranSales="urn:sales
_2017_1.transactions.webservices.netsuite.com">
      <tranSales:itemJoin xmlns:platformCommon="urn:common_2017_1.platform.webservices.netsuite
.com">
         <platformCommon:customFieldList>
            <platformCore:customField internalId="item_custitem_color" xsi:type="platformCore:S
earchColumnSelectCustomField">
               <platformCore:searchValue internalId="1" typeId="6" />
            </platformCore:customField>
         </platformCommon:customFieldList>
      </tranSales:itemJoin>
   </platformCore:searchRow>
</platformCore:searchRowList>
```

Example — Behavior with 2013.2 and Later Endpoints

The SOAP request below represents another TransactionSearchAdvanced that joins to the item record. (In this example, because the 2013.2 WSDL is being used, the “custitem_color” value is identified as the scriptId, rather than internalId.)

```
<soapenv:Body>
   <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <searchRecord xsi:type="ns6:TransactionSearchAdvanced" xmlns:ns6="urn:sales_2017_1.transa
ctions.webservices.netsuite.com">
         <ns6:criteria xsi:type="ns6:TransactionSearch">
            <ns6:basic xsi:type="ns7:TransactionSearchBasic" xmlns:ns7="urn:common_2017_1.platf
orm.webservices.netsuite.com">
               <ns7:internalId operator="anyOf" xsi:type="ns8:SearchMultiSelectField" xmlns:ns8
="urn:core_2017_1.platform.webservices.netsuite.com">
                  <ns8:searchValue internalId="67" xsi:type="ns8:RecordRef"/>
               </ns7:internalId>
           <ns7:type operator="anyOf" xsi:type="ns9:SearchEnumMultiSelectField" xmlns:ns9="urn:
core_2017_1.platform.webservices.netsuite.com">
                  <ns9:searchValue xsi:type="xsd:string">_estimate</ns9:searchValue>
           </ns7:type>
```

```
            </ns6:basic>
            </ns6:criteria>
            <ns6:columns xsi:type="ns6:TransactionSearchRow">
               <ns6:itemJoin xsi:type="ns10:ItemSearchRowBasic" xmlns:ns10="urn:common_2017_1.p
latform.webservices.netsuite.com">
                  <ns10:customFieldList xsi:type="ns11:SearchColumnCustomFieldList" xmlns:ns11=
"urn:core_2017_1.platform.webservices.netsuite.com">
                     <ns11:customField scriptId="custitem_color" xsi:type="ns11:SearchColumnSel
ectCustomField"/>
                  </ns10:customFieldList>
               </ns6:itemJoin>
            </ns6:columns>
         </searchRecord>
      </search>
</soapenv:Body>
```

In the SOAP response, the scriptId value – “custitem_color” – is unchanged. No prefix has been added.

```
<platformCore:searchRowList>
  <platformCore:searchRow xsi:type="tranSales:TransactionSearchRow" xmlns:tranSales="urn:sales_
2017_1.transactions.webservices.netsuite.com">
     <tranSales:itemJoin xmlns:platformCommon="urn:common_2017_1.platform.webservices.netsuite.
com">
        <platformCommon:customFieldList>
           <platformCore:customField internalId="211" scriptId="custitem_color" xsi:type="platf
ormCore:SearchColumnSelectCustomField">
              <platformCore:searchValue internalId="1" typeId="28"/>
           </platformCore:customField>
        </platformCommon:customFieldList>
     </tranSales:itemJoin>
  </platformCore:searchRow>
</platformCore:searchRowList>
```

Joining Through Custom Fields

If you are using 2013.2 or a later WSDL, you may notice that many *RecordType* Search objects include CustomSearchJoin. This object lets you join to records that are associated with the primary record through a custom field.

That is, if Record A includes a custom field of type list/record or multiple select, and this custom field references another record that is exposed to SuiteTalk, the 2013.2 WSDL and later WSDLs allow your search of Record A to include a join to the SearchBasic of the referenced record.

Similarly, many *RecordType* SearchRow objects now include CustomSearchRowBasic. You can use this object to include columns from the referenced record in the results of an advanced search.

See the following for more details on executing searches using CustomSearchJoin and CustomSearchRowBasic:

- Overview of CustomSearchJoin and CustomSearchRowBasic
- CustomSearch Objects Defined
- Example – Using CustomSearchJoin and CustomSearchRowBasic

- CustomSearchJoin Usage Notes

Overview of CustomSearchJoin and CustomSearchRowBasic

This topic describes some of the basics of using CustomSearchJoin and CustomSearchRowBasic.

Executing a Search

To use CustomSearchJoin, you can execute either a joined search or an advanced search.

For example, consider an employee record that has been altered to include a custom field pointing to the account record. In this case, you can search the employee record and use CustomSearchJoin to include data from the account record as part of your search criteria. You would execute your search using either of the following two objects:

- EmployeeSearch — Results returned would include the entire employee record for each employee that meets your criteria. (For general details on this type of search, see Joined Searches in Web Services.)
- EmployeeSearchAdvanced — Results returned could include any columns you specify from the employee record or the account record. (For general details on this type of search, see Advanced Searches in Web Services.) In this scenario, you would use CustomSearchRowBasic as well as CustomSearchJoin.

Sample Use Cases for CustomSearchJoin

You can use CustomSearchJoin in a couple of different scenarios.

A common case is one in which you have altered a standard record type to include a custom field that references another standard record type. This scenario includes the example where your employee record has been altered to include a custom field that points to the account record.

Additionally, you can join to and from custom record types. For example, suppose your organization uses custom record types called Customer Survey and Customer Suggestion. If Customer Survey and Customer Suggestion both include fields that reference each other, a search of Customer Survey could include a join to Customer Suggestion and vice versa.

Similarly, you can join between the following:

- A customized record type that references a standard record type.
- A standard record type that references a custom record type.

Sample Use Cases for CustomSearchRowBasic

CustomSearchRowBasic can be used in conjunction with all of the custom field usage described above for CustomSearchJoin.

Additionally, your search does not have to include the use of CustomSearchJoin in order for you to use CustomSearchRowBasic. For example, you could use EmployeeSearchAdvanced to search using criteria that is defined entirely on the employee record (and not use any joins). But in your results, you might still want to include data from the record referenced by a custom field on the employee record. You do this using CustomSearchRowBasic.

Note that CustomSearchRowBasic is available only when you are executing an advanced search.

Note: CustomSearchJoin and CustomSearchRowBasic are available only for accounts that have upgraded to 2013.2 or a later endpoint.

CustomSearch Objects Defined

For accounts that have upgraded to 2013.2 or a later endpoint, common.xsd includes two elements that support joining to and gathering result data from records that are referenced by a custom field. These objects are CustomSearchJoin and CustomSearchRowBasic. The main purpose of these objects is described in Overview of CustomSearchJoin and CustomSearchRowBasic. This topic describes how the objects are defined in the schema.

CustomSearchJoin is included in many *RecordType* Search objects. This element is defined as follows:

```
<complexType name="CustomSearchJoin">
    <sequence>
        <element name="customizationRef" type="platformCore:CustomizationRef"/>
        <element name="searchRecordBasic" type="platformCore:SearchRecordBasic"/>
    </sequence>
</complexType>
```

CustomSearchRowBasic is included in many *RecordType* SearchRow objects. This element is defined as follows:

```
<complexType name="CustomSearchRowBasic">
    <sequence>
        <element name="customizationRef" type="platformCore:CustomizationRef"/>
        <element name="searchRowBasic" type="platformCore:SearchRowBasic"/>
    </sequence>
</complexType>
```

Notice that both of these elements use CustomizationRef, which you use to identify the custom field. CustomizationRef includes five possible attributes. Of these, you can use only internal ID or script ID to identify a custom field. The other attributes (external ID, type, and name) are not applicable to custom fields.

The following example shows how you can use script ID to reference a custom field and join to AccountSearchBasic:

```
CustomizationRef field_account = new CustomizationRef(null,null,null,null,"custentity_account")
;
CustomSearchJoin customSearchJoin = new CustomSearchJoin(field_account,accountSearchBasic);
```

For a more extensive example, see Example – Using CustomSearchJoin and CustomSearchRowBasic.

Note: CustomSearchJoin and CustomSearchRowBasic are available only for accounts that have upgraded to the 2013.2 or a later endpoint.

Example – Using CustomSearchJoin and CustomSearchRowBasic

If you are using the 2013.2 WSDL or later, you have access to CustomSearchJoin and CustomSearchRowBasic. This topic shows how you might use these objects in your code. For details on the general purpose of these objects and sample use cases, see Overview of CustomSearchJoin and CustomSearchRowBasic.

The example below assumes that your employee record has been altered to include a custom field that references the account record. This sample shows how you might use EmployeeSearchAdvanced to find employee records associated with an account of a certain type and value.

```
public  void testSearch1() throws Exception{


// We are searching for employees associated with a certain type of account. Here we specify th
at account.

AccountSearchBasic accountSearchBasic = new AccountSearchBasic();
accountSearchBasic.setType(new SearchEnumMultiSelectField(new String[]{AccountType.__bank}, Sea
rchEnumMultiSelectFieldOperator.anyOf));


// Here we specify the field on the employee record that references the account record.

CustomizationRef field_account = new CustomizationRef();
field_account.setScriptId("custentity_account");


// Here we create a custom search join object.
CustomSearchJoin customSearchJoin = CustomSearchJoin(field_account, accountSearchBasic);


// Next, we pass the custom search join that we just created into an EmployeeSearch instance.

EmployeeSearch employeeSearch = new EmployeeSearch();
employeeSearch.setCustomSearchJoin(new CustomSearchJoin[] {customSearchJoin});


// Next, we specify that we want the results to include the balance of the account.

AccountSearchRowBasic accountSearchRowBasic = new AccountSearchRowBasic();
accountSearchRowBasic.setBalance(new SearchColumnDoubleField[]{new SearchColumnDoubleField()});

CustomSearchRowBasic customSearchRowBasic =new CustomSearchRowBasic(field_account,accountSearch
RowBasic);


// Here, we specify that we want the employee’s internal ID included in the results.

EmployeeSearchRowBasic employeeSearchRowBasic = new EmployeeSearchRowBasic();
employeeSearchRowBasic.setInternalId(new SearchColumnSelectField[]{new SearchColumnSelectField(
)});


// In the following snippet, we pass the specified search columns into an EmployeeSearchRow ins
tance.

EmployeeSearchRow employeeSearchRow = new EmployeeSearchRow();
employeeSearchRow.setCustomSearchJoin(new CustomSearchRowBasic[]{customSearchRowBasic});
employeeSearchRow.setBasic(employeeSearchRowBasic);
// Create the advanced search.

EmployeeSearchAdvanced employeeSearchAdvanced = new EmployeeSearchAdvanced();
employeeSearchAdvanced.setCriteria(employeeSearch);
employeeSearchAdvanced.setColumns(employeeSearchRow);

c.search(employeeSearchAdvanced);

}
```

Note: CustomSearchJoin and CustomSearchRowBasic are available only for accounts that have upgraded to the 2013.2 or a later endpoint.

CustomSearchJoin Usage Notes

If you are using the 2013.2 WSDL or later, you have access to CustomSearchJoin. Refer to the following sections to better understand the capabilities of this search object:

- Use CustomSearchJoin Only With Custom Fields
- CustomSearchJoin Connects Only With SearchBasics
- Nested Joins Are Not Allowed

Use CustomSearchJoin Only With Custom Fields

You cannot use CustomSearchJoin in conjunction with a standard field that references another record. For example, the standard employee record includes the Department field. If you decide to use CustomSearchJoin, and you identify the Department field as your custom field, the system returns an error reading “Invalid custom field reference.” You should use the standard departmentJoin that is already provided as part of the EmployeeSearch object.

CustomSearchJoin Connects Only With SearchBasics

CustomSearchJoin allows only for a basic search of the referenced record. For example, if the employee record has been altered to include a custom field that points to the account record, you can join to AccountSearchBasic – but not AccountSearch or AccountSearchAdvanced. This limitation is consistent with the capabilities of searching in the UI.

Nested Joins Are Not Allowed

You cannot use nested joins. For example, suppose your contact record includes a custom field referencing the employee record, which includes a custom field that references the account record. A search of the contact record could include a join to the employee record, but the same search could not *also* join to the account record.

Note: CustomSearchJoin and CustomSearchRowBasic are available only for accounts that have upgraded to the 2013.2 or a later endpoint.

Setting Valid Search Values

Prior to the 2008.2 endpoint, if you performed a search that included an invalid search enum filter value, you would generally still get records returned in your search. For example, if you performed a search for all customers with the Country enum search value set to "United States" (rather than the supported enum value _unitedStates), you would still get results from your search. Although the value "United States" was not recognized, you continued to get customer records. Note, however, the search results returned ALL customers in the system, since the value “United States” was invalid.

Starting with the 2008.2 endpoint, instead of ignoring any invalid search enum values, and still returning search results, NetSuite now returns 0 records and a no-match warning. Therefore, when setting search values, be sure to use the values defined in the schema.

The following is an example of what is now returned if invalid values are specified:

```
<platformCore:searchResult
xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
<platformCore:status isSuccess="true">
<platformCore:statusDetail type="WARN">
<platformCore:code>WARNING</platformCore:code>
<platformCore:message>The field country’s enum value <United States> is
invalid for this search.</platformCore:message>
</platformCore:statusDetail>
</platformCore:status>
<platformCore:totalRecords>0</platformCore:totalRecords>
<platformCore:totalPages>0</platformCore:totalPages>
<platformCore:searchId>WEBSERVICES_MSTRWLF_10212008563721605896842316_60315faa132ad</platformCo
re:searchId>
<platformCore:searchRowList/>
```

Setting the anyof, mine, or myteam Filtering Values

In SuiteTalk you can further define your search using the following filtering values:

Filter	Note
@NONE@	Equates to *anyof* (see Filtering Lists that Contain Null Values) or *unassigned* depending on the field.
@CURRENT@	Equates to *mine*. For example, use this filter to return all of your own events.
@HIERARCHY@	Equates to *my team*. For example, use this filter on a salesRep field for customer records. If you have previously defined the members of your sales team, using the @HIERARCHY@ filter will return only the customers that have worked with members of your sales team.

The following provides a SOAP sample for finding *my* events.

```
 <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <searchRecord xsi:type="ns1:CalendarEventSearchBasic"
   xmlns:ns1="urn:common_2017_1.platform.webservices.netsuite.com">
      <ns1:attendee operator="anyOf" xsi:type="ns2:SearchMultiSelectField"
   xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">
         <ns2:searchValue internalId="@CURRENT@" xsi:type="ns2:RecordRef"/>
      </ns1:attendee>
  </searchRecord>
  </search>
```

Filtering Lists that Contain Null Values

For select lists or multi selects which can have a null value, the UI supports the search criteria on these list type fields as “None Of” “-None-“, which essentially means not “Any Of” all list options. Such a search would result in all records which do NOT have this list type field as null. To accomplish this “None Of” “-None-“ search you need to set the internalId of the search key to “@None@”.

Example

To search for Customers which have a Partner associated with them in NetSuite, the SOAP would look as below for the “Partner field not null” part:

```
<ns3:partner operator="noneOf">
<ns8:searchValue internalId="@NONE@" xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.c
om"/>
</ns3:partner>
<ns3:customFieldList>
```

Java code to generate this SOAP would be:

```
// Adding search criteria where Partner is not null
RecordRef[] noPartner = new RecordRef[1];
noPartner[0] = new RecordRef();
noPartner[0].setInternalId("@NONE@");
SearchMultiSelectField partner = new SearchMultiSelectField();
partner.setOperator(SearchMultiSelectFieldOperator.noneOf);
partner.setSearchValue(noPartner);
custSearch.setPartner(partner);
```

C# code to generate this SOAP would be:

```
// Adding search criteria where Partner is not null
RecordRef[] noPartner = new RecordRef[1];
noPartner[0] = new RecordRef();
noPartner[0].internalId = "@NONE@";
SearchMultiSelectField partner = new SearchMultiSelectField();
partner.@operator = SearchMultiSelectFieldOperator.noneOf;
partner.searchValue = noPartner;
custSearch.partner = partner;
```

Searching by lastModifiedDate

This sample shows how to create a customer, and then search for the customer that was created based on a specific time frame. In this case, the sample uses the lastModifiedDate field to search “within” a couple of seconds before the customer was created and then a minute after. The search returns the record that was previously created.

Java

```
Customer c = (Customer) new TestCustomer().createMinimalRecord();
Calendar timeFrom = Calendar.getInstance();
WriteResponse wr = sessMgr.getPort().add(c);
```

```
outputResult(wr.getStatus().isIsSuccess());

Calendar timeTo = Calendar.getInstance();
timeTo.setTimeInMillis(timeTo.getTimeInMillis() + 60000);

CustomerSearch cs = new CustomerSearch();
CustomerSearchBasic csb = new CustomerSearchBasic();
SearchDateField sdf = new SearchDateField();
sdf.setOperator(SearchDateFieldOperator.within);
sdf.setSearchValue(timeFrom);
sdf.setSearchValue2(timeTo);
csb.setLastModifiedDate(sdf);

cs.setBasic(csb);

SearchResult sr = sessMgr.getWrappedPort().search(cs, this);
```

SOAP

```
<searchRecord xsi:type="ns1:CustomerSearch"
   xmlns:ns1="urn:relationships_2017_1.lists.webservices.netsuite.com">
               <ns1:basic xsi:type="ns2:CustomerSearchBasic"
   xmlns:ns2="urn:common_2017_1.platform.webservices.netsuite.com">
                  <ns2:lastModifiedDate operator="within" xsi:type="ns3:SearchDateField"
   xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">
                     <ns3:searchValue xsi:type="xsd:dateTime">2007-02-10T00:16:17.750Z</ns3:sea
rchValue>
                     <ns3:searchValue2 xsi:type="xsd:dateTime">2007-02-10T00:17:53.015Z</ns3:se
archValue2>
                  </ns2:lastModifiedDate>
               </ns1:basic>
            </searchRecord>
```

Understanding Sorting in Advanced Search

In web services, when users return a saved search that has sorting criteria specified in the saved search, the records are returned according to the specified *sorted by* order. To see the *sorted by* criteria that has been applied to a saved search, users can look at the saved search criteria in the UI.

In an ad-hoc web services search (a search in which sorted by criteria have not been set), users should be aware of the implicit sorted by order in which records are returned. This order is based on record type. Record types and the default sort by order for each record type are listed in the following table:

Note: For a list of all records associated with each type, see the help topic Web Services Supported Records.

Record Type	Default “Sorted by” Order
Entities	Name (the name of the entity)
Actvities	Event (the title of the Event)
Marketing	Campaign ID

Record Type	Default “Sorted by” Order
	Note: Promotion Code searches are sorted and returned by promotion code name.
Transactions	Date Created
Support	Number **Note:** Topic searches are sorted and returned by topic title.
File Cabinet	Name (the name of the file or folder)
Items	Name (the name of the item)
Communications (includes the Note and the Message records)	Note - sorted/returned by Author Message - sorted/returned by Message Internal ID
Website	Name
Lists	Name **Note:** Gift Certificate searches are sorted and returned by Name (From).

Request

The SearchRequest type is used for the request. It contains the following field.

Element Name	XSD Type	Notes
searchRecord	SearchRecord	The SearchRecord type is an abstract type. An instance of a type that extends SearchRecord must be used—such as CustomerSearchBasic or EventSearchBasic.

Response

The SearchResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
status	Status	The status for this search. All applicable errors or warnings will be listed within this type.
totalRecords	xsd:int	The total number of records for this search. Depending on the pageSize value, some or all the records may be returned in this response
pageSize	xsd:int	The page size for this search.
totalPages	xsd:int	The total number of pages that are part of this search.
pageIndex	xsd:int	The page index for the current set of results.
searchId	string	Returns a specific search based on its search ID.

Element Name	XSD Type	Notes
recordList	Record[]	A list of records that meet the criteria for this search. The actual records returned need to be of a type that extends the abstract type of record.
searchRowList	SearchRowList	A list of return columns that meet the criteria for this search.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Search-Related Sample Code

See the following search-related code samples:

- Basic Search Code Sample
- Joined Search Code Samples
- Advanced Search Code Samples
- Searching for a Multi-select Custom Field

Basic Search Code Sample

SOAP Request

In the following example, customer records that have an email that contains shutterfly.com are searched for. Note that you can limit the search page size at the request level (see pageSize).

```
<soap:Body>
<platformMsgs:search>
<searchRecord xsi:type="ContactSearch">
   <customerJoin xsi:type="CustomerSearchBasic">
      <email operator=”contains” xsi:type="platformCore:SearchStringField">
      <platformCore:searchValue>shutterfly.com</platformCore:searchValue>
      <email>
   <customerJoin>
</searchRecord>
</search>
</soap:Body>
```

SOAP Response

Notice that in this example, only one matching record was found. You can see that the page size was set such that if multiple records were found, only 10 would be returned at a time. The searchMore or searchNext operation could then be performed to return additional results.

```
<soapenv:Body>
<searchResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<searchResult xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
<status isSuccess="true"/>
<totalRecords>1</totalRecords>
<pageSize>10</pageSize>
<totalPages>1</totalPages>
<pageIndex>1</pageIndex>
<recordList>
   <record internalId="983" xsi:type="ns1:Customer"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ns1="urn:relationships_2017_1.lists.webservices.netsuite.com">
      <ns1:entityId>Shutter Fly</ns1:entityId>
      <ns1:isInactive>false</ns1:isInactive>
      <ns1:companyName>Shutter Fly, Inc</ns1:companyName>
      <ns1:entityStatus internalId="6"><name>LEAD-New</name>
      .
      .
      .
      <ns1:customFieldList>
         <customField internalId="165" scriptId="custentity_map" xsi:type="StringCustomFieldRef
">
            <value>http://maps.google.com</value>
         </customField>
         <customField internalId="76" scriptId="custentity_had_order_problems"
         xsi:type="BooleanCustomFieldRef"><value>false</value></customField>
      </ns1:customFieldList>
   </record>
</recordList>
</searchResult>
</searchResponse>
</soapenv:Body>
```

C#

```
 private void searchCustomer()
{
   // This operation requires a valid session
   this.login( true );

   _out.writeLn( "\nEnter search parameters" );

   // Instantiate a search object for customers. Note that the search
   // object is different from the regular record used for add and update.
   CustomerSearch custSearch = new CustomerSearch();

   // Search the customer entity id which is a string field
   _out.write( "Entity ID (press enter to skip): " );
   String nameValue = _out.readLn();
   SearchStringField entityId = null;
   if ( !nameValue.Equals( "" ) )
   {
      entityId = new SearchStringField();
      entityId.@operator = SearchStringFieldOperator.contains;
      entityId.operatorSpecified = true;
      entityId.searchValue = nameValue;
      custSearch.basic.entityId = entityId; //see note below for an alternative
   }
//Note:  You could also use CustomerSearchBasic to access entityId directly, for example:
      CustomerSearchBasic custSearchBasic = new CustomerSearchBasic();
      custSearchBasic.entityId = entityId;
   // Search the customer stage which is a list field
   _out.write( "Customer Stage (one or more nsKeys separated by commas, press enter to skip): "
 );
   String stageKeysValue = _out.readLn();
   SearchMultiSelectField stage = null;
   if ( !stageKeysValue.Equals( "" ) )
   {
      stage = new SearchMultiSelectField();
      stage.@operator = SearchMultiSelectFieldOperator.anyOf;
      stage.operatorSpecified = true;

      string [] nskeys = stageKeysValue.Split( new Char[] {','} );

      RecordRef[] recordRefs = new RecordRef[ stageKeysValue.Length ];
      for (int i=0; i<nskeys.Length; i++ )
      {
         RecordRef recordRef = new RecordRef();
         recordRef.internalId = nskeys[i];
         recordRefs[i] = recordRef;
      }
      stage.searchValue = recordRefs;
      custSearch.stage = stage;
   }

   // Search by isActive field which is a boolean
   /*
   SearchBooleanField isActive = new SearchBooleanField();
   while ( true )
   {
      _out.write( "Is active [T/F] (default is T): " );
      String upcomingStr = _out.readLn();

      if ( "T".Equals( upcomingStr.ToUpper() ) || "".Equals( upcomingStr.ToUpper() ) )
      {
         isActive.searchValue = true;
         isActive.searchValueSpecified = true;
         break;
      }
      else if ( upcomingStr.ToUpper().Equals( "F" ) )
      {
         isActive.searchValue = false;
         isActive.searchValueSpecified = true;
```

```
            break;
        }
        else
        {
            _out.writeLn( "Invalid selection" );
        }
    }
    custSearch.active = isActive;
    */
    if ( custSearch.entityId == null && custSearch.stage == null )
    {
        _out.info( "\nNo search criteria was specified. Searching for all records." );
    }
    else
    {
        _out.info(
        "\nSearching for customers with the following criteria: " +
        (entityId==null ? "" : ("\nentityID=" + custSearch.entityId.searchValue) + ",
        Operator=" + entityId.@operator.ToString()) +
        (stage==null ? "" : ("\nstage nsKeys='" + stageKeysValue + "',
        Operator=" + stage.@operator.ToString())) );
    }


    // Invoke search() web services operation
    SearchResult response = _service.search( custSearch );

    // Process response
    if ( response.status.isSuccess )
    {
        // Process the records returned in the response and print to console
        processCustomerSearchResponse( response );

        // Since pagination controls what is returned, check to see
        // if there are anymore pages to retrieve.
        searchMore( response );
    }
    else
    {
        _out.error( getStatusDetails( response.status ) );
    }
}
```

Java

```
public void searchCustomer() throws RemoteException,
ExceededUsageLimitFault, UnexpectedErrorFault, InvalidSessionFault,
ExceededRecordCountFault {
    // This operation requires a valid session
    this.login(true);

    _console.writeLn("\nEnter search parameters");

    // Instantiate a search object for customers. Note that the search
    // object is different from the regular record used for add and update.
    CustomerSearch custSearch = new CustomerSearch();

    // Search the customer entity id which is a string field
    _console.write("Entity ID (press enter to skip): ");
    String nameValue = _console.readLn();
    SearchStringField entityId = null;
    if (!nameValue.equals("")) {
       entityId = new SearchStringField();
       entityId.setOperator(SearchStringFieldOperator.contains);
       entityId.setSearchValue(nameValue);
       custSearch.setEntityId(entityId);
    }

    // Search the customer stage which is a list field
    _console
    .write("Customer Stage (one or more nsKeys separated by commas, press enter to skip): ");
    String stageKeysValue = _console.readLn();
    SearchMultiSelectField stage = null;
    if (!stageKeysValue.equals("")) {
       stage = new SearchMultiSelectField();
       stage.setOperator(SearchMultiSelectFieldOperator.anyOf);

       String[] nskeys = stageKeysValue.split(",");

       RecordRef[] recordRefs = new RecordRef[stageKeysValue.length()];
       for (int i = 0; i < nskeys.length; i++) {
          RecordRef recordRef = new RecordRef();
          recordRef.setInternalId(nskeys[i]);
          recordRefs[i] = recordRef;
       }
       stage.setSearchValue(recordRefs);
       custSearch.setStage(stage);
    }
    if (custSearch.getEntityId() == null && custSearch.getStage() == null) {
       _console
       .info("\nNo search criteria was specified. Searching for all records.");
    } else {
       _console
       .info("\nSearching for customers with the following criteria: "
       + (entityId == null ? ""
       : ("\nentityId=" + custSearch
       .getEntityId().getSearchValue())
       + ", Operator="
       + entityId.getOperator().toString())
       + (stage == null ? "" : ("\nstage nsKeys='"
       + stageKeysValue + "', Operator=" + stage
       .getOperator().toString())));
    }

    // Set page size for number of records to be returned in search
    // response

    // Invoke search() web services operation
    SearchResult result = _port.search(custSearch);
```

```
   // Process result
   if (result.getStatus().isIsSuccess()) {
      // Process the records returned in the result and print to console
      processCustomerSearchResponse(result);

      // Since pagination controls what is returned, check to see
      // if there are anymore pages to retrieve.
      searchMore(result);
   } else {
      _console.error(getStatusDetails(result.getStatus()));
   }
}
```

Joined Search Code Samples

Sample 1 — Customer Search With Contact Join

This sample shows how to execute a customer search in which the contact email address is specified as the search criteria.

Java

```
public void contactSearch_with_CustomerJoin() throws Exception {
    ContactSearch cs = new ContactSearch();
    ContactSearchBasic contactSearchBasic = new ContactSearchBasic();
    contactSearchBasic.setEmail(new SearchStringField("contact@email.com", SearchStringFieldOpe
rator.is));
    CustomerSearchBasic customerSearchBasic = new CustomerSearchBasic();
    customerSearchBasic.setEntityId(new SearchStringField("My Customer", SearchStringFieldOpera
tor.is));
    cs.setBasic(contactSearchBasic);
    cs.setCustomerJoin(customerSearchBasic);
    sessMgr.getWrappedPort().search(cs, this);
}
```

SOAP

```
<search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
       <searchRecord xsi:type="ns4:ContactSearch"
      xmlns:ns4="urn:relationships_2017_1.lists.webservices.netsuite.com">
               <ns4:basic xsi:type="ns5:ContactSearchBasic"
      xmlns:ns5="urn:common_2017_1.platform.webservices.netsuite.com">
                  <ns5:email operator="is" xsi:type="ns6:SearchStringField"
      xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.com">
                     <ns6:searchValue xsi:type="xsd:string">contact@email.com</ns6:searchValue>

                  </ns5:email>
               </ns4:basic>
               <ns4:customerJoin xsi:type="ns7:CustomerSearchBasic"
      xmlns:ns7="urn:common_2017_1.platform.webservices.netsuite.com">
                  <ns7:entityId operator="is" xsi:type="ns8:SearchStringField"
      xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com">
                     <ns8:searchValue xsi:type="xsd:string">My Customer</ns8:searchValue>
                  </ns7:entityId>
               </ns4:customerJoin>
            </searchRecord>
         </search>
```

Sample 2 — Contact Search With Customer Join

The following sample shows how to return an associated joined list of records. In this case, all contacts associated with customers of internalId 1, 2 and 3 are returned.

Java

```
   RecordRef[] rr = new RecordRef[]{new RecordRef("1", RecordType.customer),
   new RecordRef("2", RecordType.customer), new RecordRef("3", RecordType.customer)};
   CustomerSearchBasic customerSearchBasic = new CustomerSearchBasic();
   customerSearchBasic.setInternalId(new SearchMultiSelectField(rr,
   SearchMultiSelectFieldOperator.anyOf));
   ContactSearch contactSearch = new ContactSearch();
   contactSearch.setCustomerJoin(customerSearchBasic);
```

SOAP

```
<soapenv:Body>
   <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <searchRecord xsi:type="ns1:ContactSearch"
      xmlns:ns1="urn:relationships_2017_1.lists.webservices.netsuite.com">
      <ns1:customerJoin xsi:type="ns2:CustomerSearchBasic"
         xmlns:ns2="urn:common_2017_1.platform.webservices.netsuite.com">
         <ns2:internalId operator="anyOf" xsi:type="ns3:SearchMultiSelectField"
            xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">
            <ns3:searchValue internalId="1" type="customer"
             xsi:type="ns3:RecordRef"/>
            <ns3:searchValue internalId="2" type="customer"
             xsi:type="ns3:RecordRef"/>
            <ns3:searchValue internalId="3" type="customer"
             xsi:type="ns3:RecordRef"/>
         </ns2:internalId>
      </ns1:customerJoin>
   </searchRecord>
   </search>
</soapenv:Body>
```

Sample 3 — Item Search With Pricing Join

The following sample shows how to search for all items that have a price level of 10.00.

C#

```
private void myItemSearch()
```

```
{

   ItemSearchBasic myItemSearchBasic = new ItemSearchBasic();

   SearchEnumMultiSelectField myEnum = new SearchEnumMultiSelectField();
   myEnum.@operator = SearchEnumMultiSelectFieldOperator.anyOf;
   myEnum.operatorSpecified = true;
   String[] searchStringArray = new String[1];
   searchStringArray[0] = "_inventoryItem";
   myEnum.operatorSpecified = true;
   myEnum.searchValue = searchStringArray;

   myItemSearchBasic.type = myEnum;


   PricingSearchBasic myPricingSearchBasic = new PricingSearchBasic();

   SearchDoubleField rateValue = new SearchDoubleField();
   rateValue.@operator = SearchDoubleFieldOperator.equalTo;
   rateValue.operatorSpecified = true;
   rateValue.searchValue = 10.00;
   rateValue.searchValueSpecified = true;

   myPricingSearchBasic.rate = rateValue;


   ItemSearch myItemSearch = new ItemSearch();
   myItemSearch.basic = myItemSearchBasic;
   myItemSearch.pricingJoin = myPricingSearchBasic;

   SearchResult searchResult = _service.search(myItemSearch);
}
```

SOAP Request

```
<soap:Body>
   <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <searchRecord xmlns:q1="urn:accounting_2017_1.lists.webservices.netsuite.com" xsi:type="q
1:ItemSearch">
         <q1:basic>
            <type operator="anyOf" xmlns="urn:common_2017_1.platform.webservices.netsuite.com">

               <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">_inventor
yItem</searchValue>
            </type>
         </q1:basic>
        <q1:pricingJoin>
            <rate operator="equalTo" xmlns="urn:common_2017_1.platform.webservices.netsuite.com
">
               <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">10</searc
hValue>
            </rate>
         </q1:pricingJoin>
      </searchRecord>
   </search>
```

```
</soap:Body>
```

Advanced Search Code Samples

The following advanced search samples are provided.

1. Perform a search that includes a saved search ID.
2. Perform a search with a defined page size.
3. Perform a search in which you override search columns in a saved search.
4. Perform a search in which you specify additional saved search criteria.
5. Programmatically manipulate data returned by a search

Perform a search that includes a saved search ID.

C#

```
// Create a service
NetSuiteService nss = new NetSuiteService();

// Perform the search. Note that you can get the saved search ID using
// either getSavedSearch() or through the UI
TransactionSearchAdvanced tsa1 = new TransactionSearchAdvanced();
tsa1.savedSearchId="57";  //substitute your own saved search internal ID
nss.search(tsa1);
```

SOAP Request

```
<search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<searchRecord xmlns:q1="urn:sales_2017_1.transactions.webservices.netsuite.com"
   xsi:type="q1:TransactionSearchAdvanced" savedSearchId="57" />
</search>
```

SOAP Response

```
// The default is to return search rows
<platformCore:searchRow xsi:type="tranSales:TransactionSearchRow"
   xmlns:tranSales="urn:sales_2017_1.transactions.webservices.netsuite.com">
<tranSales:basic xmlns:platformCommon="urn:common_2017_1.platform.webservices.netsuite.com">
<platformCommon:account>
<platformCore:searchValue internalId="125"/>
</platformCommon:account>
<platformCommon:amount>
<platformCore:searchValue>12481.9</platformCore:searchValue>
</platformCommon:amount>
<platformCommon:entity>
<platformCore:searchValue internalId="78"/>
</platformCommon:entity>
<platformCommon:mainline>
<platformCore:searchValue>false</platformCore:searchValue>
</platformCommon:mainline>
```

```
<platformCommon:number>
<platformCore:searchValue>102</platformCore:searchValue>
</platformCommon:number>
<platformCommon:postingPeriod>
<platformCore:searchValue internalId="72"/>
</platformCommon:postingPeriod>
<platformCommon:tranDate>
<platformCore:searchValue>2005-02-10T00:00:00.000-08:00</platformCore:searchValue>
</platformCommon:tranDate>
<platformCommon:tranId>
<platformCore:searchValue>102</platformCore:searchValue>
</platformCommon:tranId>
<platformCommon:type>
<platformCore:searchValue internalId="SalesOrd"/>
</platformCommon:type>
</tranSales:basic>
</platformCore:searchRow>
```

Perform a search with a defined page size.

This sample performs the same search as the previous sample, however, you are specifying the number of results that are returned per page.

C#

```
TransactionSearchAdvanced tsa2 = new TransactionSearchAdvanced();
tsa2.savedSearchId="57";  //substitute your own saved search internal ID

// Set search preference to return search columns
SearchPreferences sp = new SearchPreferences();
sp.pageSizeSpecified = true;
sp.pageSize = 10;
nss.searchPreferences = sp;
nss.search(tsa2);
```

Perform a search in which you override search columns in a saved search.

In this sample you are returning a specific saved search (57), and then overriding the search return columns in the saved search and specifying your own search return columns.

C#

```
TransactionSearchAdvanced tsa3 = new TransactionSearchAdvanced();
tsa3.savedSearchId="57";  //substitute your own saved search internal ID

// Set search preference to return search columns
// Already defined returnSearchColumns
//SearchPreferences sp = new SearchPreferences();
//sp.returnSearchColumns = true;
//nss.searchPreferences = sp;

// Instantiate SearchColumn object
```

```
TransactionSearchRow tsr = new TransactionSearchRow();
TransactionSearchRowBasic tsrb = new TransactionSearchRowBasic();

// return internId
SearchColumnSelectField [] orderIdCols = new SearchColumnSelectField[1];
SearchColumnSelectField orderIdCol = new SearchColumnSelectField();
orderIdCol.customLabel = "Sales Order ID"; // Define a custom label
orderIdCols[0] = orderIdCol;
tsrb.internalId = orderIdCols;

SearchColumnDateField [] tranDateCols = new SearchColumnDateField[1];
SearchColumnDateField tranDateCol = new SearchColumnDateField();
tranDateCols[0] = tranDateCol;
tsrb.tranDate = tranDateCols;
SearchColumnBooleanField [] isFulfilledCols = new SearchColumnBooleanField[1];
SearchColumnBooleanField isFulfilledCol = new SearchColumnBooleanField();
isFulfilledCol.customLabel = "Order Fulfilled"; // Define a custom label
isFulfilledCols[0] = isFulfilledCol;
tsrb.shipRecvStatus = isFulfilledCols;

tsr.basic = tsrb;
tsa3.columns = tsr;
nss.search(tsa3);
```

SOAP Request

```
<search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
               <searchRecord xmlns:q1="urn:sales_2017_1.transactions.webservices.netsuite.com"
      xsi:type="q1:TransactionSearchAdvanced" savedSearchId="57">
                  <q1:columns>
                     <q1:basic>
                        <internalId xmlns="urn:common_2017_1.platform.webservices.netsuite.com"
>
                           <customLabel xmlns="urn:core_2017_1.platform.webservices.netsuite.co
m">Sales Order ID
         </customLabel>
                        </internalId>
                        <shipRecvStatus xmlns="urn:common_2017_1.platform.webservices.netsuite.
com">
                           <customLabel xmlns="urn:core_2017_1.platform.webservices.netsuite.co
m">Order Fulfilled
         </customLabel>
                        </shipRecvStatus>
                        <tranDate xmlns="urn:common_2017_1.platform.webservices.netsuite.com" /
>
                     </q1:basic>
                  </q1:columns>
               </searchRecord>
            </search>
```

SOAP Response

```
<platformCore:searchRow xsi:type="tranSales:TransactionSearchRow"
   xmlns:tranSales="urn:sales_2017_1.transactions.webservices.netsuite.com">
```

```
        <tranSales:basic xmlns:platformCommon="urn:common_2017_1.platform.webservices.netsuite.
com">
                     <platformCommon:internalId>
                     <platformCore:searchValue internalId="117"/>
                    <platformCore:customLabel>Sales Order ID</platformCore:customLabel>
                    </platformCommon:internalId>
                    <platformCommon:tranDate>
                    <platformCore:searchValue>2005-04-23T00:00:00.000-07:00</platformCore:searc
hValue>
                    </platformCommon:tranDate>
                     </tranSales:basic>
                     </platformCore:searchRow>
```

Perform a search in which you specify additional saved search criteria.

The next sample shows how to take a saved search and specify additional search filter criteria. The additional search critera will not override the criteria already defined in the saved search.

C#

```
TransactionSearchAdvanced tsa4 = new TransactionSearchAdvanced();
tsa4.savedSearchId="57";

TransactionSearch ts = new TransactionSearch();
TransactionSearchBasic tsb = new TransactionSearchBasic();
// condition 1: on SO only
SearchEnumMultiSelectField semsfTranType = new SearchEnumMultiSelectField();
semsfTranType.operatorSpecified = true;
semsfTranType.@operator = SearchEnumMultiSelectFieldOperator.anyOf;
String [] tranTypes = new String[1];
String tranType = "_salesOrder";
tranTypes[0] = tranType;
semsfTranType.searchValue = tranTypes;
tsb.type = semsfTranType;

// condition 2: tranId contains 182
SearchStringField sfTranId = new SearchStringField();
sfTranId.searchValue = "182"; // tranId contains 11
sfTranId.@operator = SearchStringFieldOperator.contains;
sfTranId.operatorSpecified = true;
tsb.tranId=sfTranId;

// condition 3: Is MultiShipping Routes enabled
SearchBooleanField sbfShipLineEnabled = new SearchBooleanField();
sbfShipLineEnabled.searchValue = true;
sbfShipLineEnabled.searchValueSpecified = true;
tsb.shipping = sbfShipLineEnabled;

// condition 4: Open SO
SearchBooleanField sbfTranStatus = new SearchBooleanField();
sbfTranStatus.searchValue = true;
sbfTranStatus.searchValueSpecified = true;
tsb.shipRecvStatusLine = sbfTranStatus;
```

```
ts.basic = tsb;
tsa4.criteria = ts;
tsa4.columns = tsr; //note - columns previously defined above.
nss.search(tsa4);
```

SOAP Request

```
<searchRecord xmlns:q1="urn:sales_2017_1.transactions.webservices.netsuite.com" xsi:type="q1:Tr
ansactionSearchAdvanced">
               <q1:criteria>
                  <q1:basic>
                     <tranId operator="contains" xmlns="urn:common_2017_1.platform.webservices.
netsuite.com">
                        <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
182</searchValue>
                     </tranId>
                     <type operator="anyOf" xmlns="urn:common_2017_1.platform.webservices.netsu
ite.com">
                        <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
_salesOrder
         </searchValue>
                     </type>
                  </q1:basic>
               </q1:criteria>
               <q1:columns>
                  <q1:basic>
                     <internalId xmlns="urn:common_2017_1.platform.webservices.netsuite.com">
                        <customLabel xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
Sales Order ID
         </customLabel>
                     </internalId>
                     <shipRecvStatus xmlns="urn:common_2017_1.platform.webservices.netsuite.com
">
                        <customLabel xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
Order Fulfilled
         </customLabel>
                     </shipRecvStatus>
                     <shipRecvStatusLine xmlns="urn:common_2017_1.platform.webservices.netsuite
.com">
                        <customLabel xmlns="urn:core_2017_1.platform.webservices.netsuite.com">

         Order Line Fulfilled</customLabel>
                     </shipRecvStatusLine>
                     <tranDate xmlns="urn:common_2017_1.platform.webservices.netsuite.com" />
                     <tranId xmlns="urn:common_2017_1.platform.webservices.netsuite.com" />
                  </q1:basic>
               </q1:columns>
            </searchRecord>
```

SOAP Request (with Shipping and ShipRecvStatusLine criteria)

```
<searchRecord xmlns:q1="urn:sales_2017_1.transactions.webservices.netsuite.com"
   xsi:type="q1:TransactionSearchAdvanced">
               <q1:criteria>
```

```
                   <q1:basic>
                      <shipRecvStatus xmlns="urn:common_2017_1.platform.webservices.netsuite.com
">
                         <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
true</searchValue>
                      </shipRecvStatus>
                      <tranId operator="contains" xmlns="urn:common_2017_1.platform.webservices.
netsuite.com">
                         <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
11</searchValue>
                      </tranId>
                      <type operator="anyOf" xmlns="urn:common_2017_1.platform.webservices.netsu
ite.com">
                         <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
_salesOrder
         </searchValue>
                      </type>
                   </q1:basic>
                </q1:criteria>
                <q1:columns>
                   <q1:basic>
                      <internalId xmlns="urn:common_2017_1.platform.webservices.netsuite.com">
                         <customLabel xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
Sales Order ID
         </customLabel>
                      </internalId>
                      <shipRecvStatus xmlns="urn:common_2017_1.platform.webservices.netsuite.com
">
                         <customLabel xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
Order Fulfilled
         </customLabel>
                      </shipRecvStatus>
                      <tranDate xmlns="urn:common_2017_1.platform.webservices.netsuite.com" />
                   </q1:basic>
                </q1:columns>
             </searchRecord>
```

SOAP Response

```
 <platformCore:searchRow xsi:type="tranSales:TransactionSearchRow" xmlns:tranSales="urn:sales_2
017_1.transactions.webservices.netsuite.com">
    <tranSales:basic xmlns:platformCommon="urn:common_2017_1.platform.webservices.netsuite.com"
>
                         <platformCommon:internalId>
                            <platformCore:searchValue internalId="986"/>
                            <platformCore:customLabel>Sales Order ID</platformCore:customLabel>
                         </platformCommon:internalId>
                         <platformCommon:shipRecvStatusLine>
                            <platformCore:searchValue>true</platformCore:searchValue>
                            <platformCore:customLabel>Order Line Fulfilled</platformCore:customL
abel>
                         </platformCommon:shipRecvStatusLine>
                         <platformCommon:tranDate>
                            <platformCore:searchValue>2008-09-11T00:00:00.000-07:00</platformCor
e:searchValue>
```

```
                        </platformCommon:tranDate>
                    </tranSales:basic>
                </platformCore:searchRow>
```

Programmatically manipulate data returned by a search

Java

```
public void searchCustomerBySavedSearch() throws RemoteException {
        this.login(true);

        CustomerSearchAdvanced searchRecord = new CustomerSearchAdvanced();
        searchRecord.setSavedSearchId("26"); // A saved customer search

        SearchResult result = _port.search(searchRecord);

        if( result.getTotalRecords() > 0 ) {
                // retain the search ID  to get more pages
                String sSearchId = result.getSearchId();

                SearchRowList rowList = result.getSearchRowList();

                processRowList(rowList);

                int iNumPages = result.getTotalPages();

                for ( int i=2; i<=iNumPages; i++) {
                        result = _port.searchMoreWithId(sSearchId, i);
                        rowList = result.getSearchRowList();

                        processRowList(rowList);
                }
                }
}

public void processRowList(SearchRowList rowList) {
        for (int i=0; i<rowList.getSearchRow().length; i++) {
                CustomerSearchRow row = (CustomerSearchRow) rowList.getSearchRow(i);
                CustomerSearchRowBasic basic = row.getBasic();

                SearchColumnStringField companyName = basic.getCompanyName(0);

                _console.write("Company Name: "+companyName.getSearchValue());
                if (basic.getEmail(0).getSearchValue() != null) {
                        String email = basic.getEmail(0).getSearchValue();
                        _console.write("\tEmail: "+email);
                }
                if (basic.getPhone(0).getSearchValue() != null) {
                        String phone = basic.getPhone(0).getSearchValue();
                        _console.write("\tPhone: "+phone);
                }
                _console.writeLn("\n");
        }
}
```

Searching for a Multi-select Custom Field

In the following code sample, the results for the custom transaction field custcolcolumnname are returned.

Java

```
   // transaction search by custom column field
   TransactionSearchBasic transactionSearch = new TransactionSearchBasic();

   SearchCustomFieldList searchCustomFieldList = new SearchCustomFieldList();
   transactionSearch.setCustomFieldList(searchCustomFieldList);

   // make the multiselectsearch
   SearchMultiSelectCustomField searchMultiSelectCustomField = new SearchMultiSelectCustomField
();

   ListOrRecordRef listOrRecordRef = new ListOrRecordRef();
   listOrRecordRef.setInternalId("3"); // the internal id of the custom list value
   listOrRecordRef.setType("1"); // your custom list typeId

   searchCustomFieldList.setCustomField(new SearchCustomField[]{searchMultiSelectCustomField});


   // make the search expression
   searchMultiSelectCustomField.setscriptId("custcolcolumnname"); //the name of the tx custom c
olumn
   searchMultiSelectCustomField.setOperator(SearchMultiSelectFieldOperator.anyOf);

   searchMultiSelectCustomField.setSearchValue(new ListOrRecordRef[] {listOrRecordRef});

   SearchResult sr = _port.search(transactionSearch);
```

C#

```
private void searchForMultiSelectCustomField()

        {
            if (_isAuthenticated)
            {
                _out.info("\nExecuting  search ..... \n");
                // transaction search by custom column field
                TransactionSearch transactionSearch = new TransactionSearch();
                TransactionSearchBasic transactionSearchBasic = new TransactionSearchBasic();

                //Java - the SearchCustomFieldList is not used.
                //SearchCustomFieldList searchCustomFieldList = new SearchCustomFieldList();
                //transactionSearch.setCustomFieldList(searchCustomFieldList);
                SearchMultiSelectCustomField searchMultiSelectCustomField = new
            SearchMultiSelectCustomField();

                // make the search expression
               //the name of the transaction custom column
        searchMultiSelectCustomField.scriptId = "custbody_multi_select";
                searchMultiSelectCustomField.@operator = SearchMultiSelectFieldOperator.anyOf;
                searchMultiSelectCustomField.operatorSpecified = true;
                //custom list called colors with typei id  1, values blue - internalid 1, green
 - id2 etc
                //we are looking for transactions which have transaction body field
         //of type multi select set to color blue
                ListOrRecordRef listOrRecordRef = new ListOrRecordRef();
                listOrRecordRef.internalId = "3";
                listOrRecordRef.typeId = "1";

                searchMultiSelectCustomField.searchValue = new ListOrRecordRef[] { listOrRecord
Ref };
                SearchCustomField[] searchCustomFieldList = new SearchCustomField[] {
     searchMultiSelectCustomField };

                //Java
                //searchCustomFieldList.setCustomField(new SearchCustomField[]{searchMultiSelec
tCustomField});
                transactionSearchBasic.customFieldList = searchCustomFieldList;
                transactionSearch.basic = transactionSearchBasic;
                SearchResult searchRes = _service.search(transactionSearch);
                _out.info("\nSearch Result contains " + searchRes.totalRecords + " record(s) \n
");
            }
            else
            {
                _out.info(
                    "\nCannot call  search operation  because there is no active session. " +
                    "You must be first logged on before attempting to call saved search.\n");
            }
        }
```

searchMore

The seachMore operation is used to retrieve more records after an initial search operation.

Important: Users who authenticate to NetSuite through login can use either searchMore or searchMoreWithId to paginate through search results. Users who authenticate to NetSuite by providing user credentials in the header of their SOAP requests can use **only** searchMoreWithId to paginate through search results, since searchMore requires an active session.

Note: (See Request-Level Credentials for information on request-level-credential authentication.)

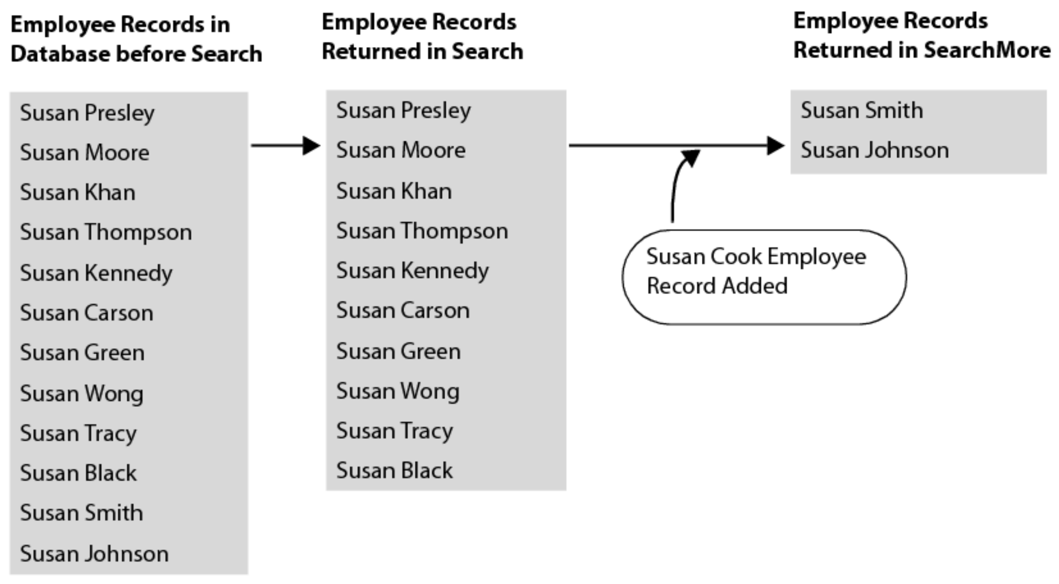
The results returned in a searchMore operation reflect the records in the next segment as defined in the **original** search operation. Therefore, if a record is deleted before all records have been returned, the total number of records may differ from the total record count returned in the search operation — the deleted record is not returned. If a record is **added**, that record is not returned in subsequent searchMore operations. However, the data in each record returned is current such that if a change to a record occurs in between the time of the original search operation and the searchMore operation, the updated data is returned.

Example

Suppose you submit the following request for all employees whose firstName contains 'Susan' and set the pagesize to 10 records:

```
<soap:Header>
   <platformMsgs:searchPreferences>
      <platformMsgs:bodyFieldsOnly>true</platformMsgs:bodyFieldsOnly>
      <platformMsgs:pageSize>10</platformMsgs:pageSize>
   </platformMsgs:searchPreferences>
</soap:Header>
<soap:Body>
   <platformMsgs:search>
      <platformMsgs:searchRecord xsi:type="listEmp:EmployeeSearch">
         <listEmp:firstName operator="contains" xsi:type="platformCore:SearchStringField">
            <platformCore:searchValue>Susan</platformCore:searchValue>
         </listEmp:firstName>
      </platformMsgs:searchRecord>
   </platformMsgs:search>
</soap:Body>
```

As illustrated in the following figure, if the database contains 12 matches at the time of the initial Search, the Search result would provide the first 10 records as defined in the pageSize preference. A subsequent SearchMore operation would return only the remaining two records from the original search result even if another employee record, Susan Cook, is added prior to the SearchMore request being submitted.



Request

The SearchMoreRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
pageIndex	SearchRecord	An index that specifies which page in the search to return. If it is not provided, the next page is returned.

Response

The SearchMoreResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
status	Status	The status for this search. All applicable errors or warnings will be listed within this type.
totalRecords	xsd:int	The total number of records for this search. Depending on the pageSize value, some or all the records may be returned in this response
pageSize	xsd:int	The page size for this search.
totalPages	xsd:int	The total number of pages that are part of this search.
pageIndex	xsd:int	The page index for the current set of results.
recordList	Record[]	A list of records that meet the criteria for this search. The actual records returned need to be of a type that extends the abstract type of record.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidCredentialsFault
- InvalidSessionFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In this example, page 2 of the search result set is requested.

```
<soap:Body>
<platformMsgs:searchMore>
   <platformMsgs:pageIndex>2</platformMsgs:pageIndex>
</platformMsgs:searchMore>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
<searchMoreResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
```

```
<searchResult xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
<status isSuccess="true"/>
<totalRecords>93</totalRecords>
<pageSize>10</pageSize>
<totalPages>10</totalPages>
<pageIndex>2</pageIndex>
<recordList>
   <record internalId="80" xsi:type="ns1:Customer"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ns1="urn:relationships_2017_1.lists.webservices.netsuite.com">
      <ns1:entityId>Jackson Alexander</ns1:entityId>
      <ns1:isInactive>false</ns1:isInactive>
      .
      ...[more fields]
      .
      <ns1:customFieldList>
         <customField internalId="custentity_map" xsi:type="StringCustomFieldRef">
            <value>http://maps.google.com</value>
         </customField>
      </ns1:customFieldList>
   </record>
   <record internalId="227" xsi:type="ns2:Customer"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ns2="urn:relationships_2017_1.lists.webservices.netsuite.com">
      <ns2:entityId>Seena Thomas</ns2:entityId>
      <ns2:isInactive>false</ns2:isInactive>
      <ns2:companyName>Seena Thom Inc.</ns2:companyName>
      .
      ...[more fields]
      .
      <ns2:customFieldList>
         <customField internalId="custentity_map" xsi:type="StringCustomFieldRef">
            <value>http://maps.google.com</value>
         </customField>
      </ns2:customFieldList>
   </record>
   .
   ...[more records]
   .
</recordList>
</searchResult>
</searchMoreResponse>
</soapenv:Body>
```

C#

```
private void searchMore( SearchResult response )
{
   //SearchResponse response;
   bool isGetAllPages = false;

   // Keep getting pages until there are no more pages to get
   while ( response.totalRecords > (response.pageSize * response.pageIndex) )
   {
      if ( !isGetAllPages )
      {
         // Only continue if user wants to get the next page
         _out.write( "\nThere are more search results. Would you like to get the next page for

         this search? (A/Y/N): " );
         String userResponse = _out.readLn().ToUpper();
         if ( String.Equals( userResponse, "N" ) )
         {
            break;
         }
         else if ( String.Equals( userResponse, "A" ) )
         {
            isGetAllPages = true;
         }
      }

      // Invoke searchMore() operation
      response = _service.searchMore( response.pageIndex + 1 );

      // Process response
      if ( response.status.isSuccess )
      {
         processCustomerSearchResponse( response );
      }
      else
      {
         _out.error( getStatusDetails( response.status ) );
      }
   }
}
```

Java

```
public void searchMore(SearchResult result) throws RemoteException,
ExceededUsageLimitFault, UnexpectedErrorFault, InvalidSessionFault,
ExceededRecordCountFault {
   // SearchResult response;
   boolean isGetAllPages = false;

   // Keep getting pages until there are no more pages to get
   while (result.getPageSize() != null
   && result.getPageIndex() != null
   && (result.getTotalRecords().intValue() > (result.getPageSize()
   .intValue() * result.getPageIndex().intValue()))) {
      if (!isGetAllPages) {
         // Only continue if user wants to get the next page
         _console
         .write("\nThere are more search results. Would you like to get the next page for
          this search? (A/Y/N):");
         String userResponse = _console.readLn().toUpperCase();
         if ("N".equals(userResponse)) {
            break;
         } else if ("A".equals(userResponse)) {
            isGetAllPages = true;
         }
      }

      // Invoke searchMore() operation
      result = _port.searchMore(result.getPageIndex().intValue() + 1);

      // Process result
      if (result.getStatus().isIsSuccess()) {
         processCustomerSearchResponse(result);
      } else {
         _console.error(getStatusDetails(result.getStatus()));
      }
   }
}
```

searchMoreWithId

Users who authenticate to NetSuite by providing their credentials in the SOAP header of their requests **must** use searchMoreWithId to retrieve search results that span multiple pages. They cannot use searchMore or searchNext, since both operations require an active session. (For information on request-level-credential authentication, see Request-Level Credentials.)

Note: Users who authenticate to NetSuite through login can use searchMore or searchMoreWithId to paginate through search results.

The searchMoreWithId operation allows users to reference a specific search result set by its searchId, a parameter included in all search results. As with searchMore, users must set the pageIndex value to specify which page in the search to return.

Usage Notes

Consider the following information when you use the searchMoreWithId operation.

- When you paginate through the search results, the searchMoreWithId operation triggers a new search for the page specified by the pageIndex value. If records included in the search result set are updated while you go through the result pages, and they do not match the earlier search criteria anymore, they will be omitted from the results displayed on that page you specified. To avoid this, you need to ensure that records are not modified while you are calling a search operation.
- Note that a particular entity is allowed to have two search IDs alive at a time. The oldest ID is expunged if a third is created. For SuiteCloud Plus users, a maximum of 20 search IDs are stored for a single SuiteCloud Plus user (two IDs per session x ten current logins). For information on the SuiteCloud Plus license, see Enabling Web Services Concurrent Users with SuiteCloud Plus.
- Search IDs expire if they have not been used within 15 minutes after their creation. Passing an expired or invalid searchId will return search results with a “failed” status and StatusDetailCode=INVALID_JOB_ID.
- There is no async equivalent for this operation.

Request

The SearchMoreWithIdRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
searchId	string	The search result ID.
pageIndex	int	An index that specifies which page in the search to return.

Response

The SearchMoreWithIdResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
status	Status	The status for this search. All applicable errors or warnings will be listed within this type.
totalRecords	xsd:int	The total number of records for this search. Depending on the pageSize value, some or all the records may be returned in this response
pageSize	xsd:int	The page size for this search.
totalPages	xsd:int	The total number of pages that are part of this search.
pageIndex	xsd:int	The page index for the current set of results.
searchId	string	Returns a specific search based on its search ID.
recordList	Record[]	A list of records that meet the criteria for this search. The actual records returned need to be of a type that extends the abstract type of record.
searchRowList	SearchRowList	A list of return columns that meet the criteria for this search.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

The following samples show the SOAP for the initial search as well as the SOAP for subsequent searchMoreWithId operation used to specify the next page of the search. The ID returned in the initial search is specified as the searchId when executing searchMoreWithId.

Note: Prefix-to-namespace mappings have been omitted for readability.

SOAP Request for Initial Search

```
<soapenv:Envelope>
    <soapenv:Header>
        <platformMsgs:searchPreferences>
            <platformMsgs:bodyFieldsOnly>true</platformMsgs:bodyFieldsOnly>
            <platformMsgs:pageSize>100</platformMsgs:pageSize>
        </platformMsgs:searchPreferences>
        <platformMsgs:passport>
            <platformCore:email>jdoe@netsuite.com</platformCore:email>
            <platformCore:password>mypassword</platformCore:password>
            <platformCore:account>000034</platformCore:account>
            <platformCore:role internalId="3"/>
        </platformMsgs:passport>
    </soapenv:Header>
    <soapenv:Body>
        <platformMsgs:search>
            <searchRecord xsi:type="platformCommon:ContactSearchBasic">
                <platformCommon:city operator="is" xsi:type="platformCore:SearchStringField">
                    <platformCore:searchValue xsi:type="xsd:string">San Francisco</platformCore
:searchValue>
                </platformCommon:city>
            </searchRecord>
        </platformMsgs:search>
    </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response for Initial Search

```
<soapenv:Envelope>
    <soapenv:Header>
        <platformMsgs:documentInfo>
            <platformMsgs:nsId>WEBSERVICES_528736_07012008543995006307049233_d53ef4d2273b15<
      platformMsgs:nsId>
        </platformMsgs:documentInfo>
    </soapenv:Header>
    <soapenv:Body>
        <platformMsgs:searchResponse>
          <platformCore:searchResult>
          <platformCore:status isSuccess="true"/>
          <platformCore:totalRecords>231</platformCore:totalRecords>
          <platformCore:pageSize>100</platformCore:pageSize>
          <platformCore:totalPages>3</platformCore:totalPages>
          <platformCore:pageIndex>1</platformCore:pageIndex>
            <platformCore: searchId > WEBSERVICES_528736_07012008543995006307049233_d53ef4d2273
b15
      <platformCore: searchId >
           <platformCore:recordList>
```

```
            <platformCore:recordList>
                  <platformCore:record internalId="4" externalId="entity-4" xsi:type="listRel:Co
ntact">
                       <listRel:entityId>john</listRel:entityId>
                       <listRel:firstName>John</listRel:firstName>
                         .....
                     </platformCore:record>
                 </platformCore:recordList>
             </platformCore:searchResult>
         </searchMoreWithIdResponse>
     </soapenv:Body>
</soapenv:Envelope>
```

SOAP Request for Getting the Next Page Using searchMoreWithId

```
<soapenv:Envelope>
    <soapenv:Header>
        <platformMsgs:searchPreferences>
            <platformMsgs:bodyFieldsOnly>true</platformMsgs:bodyFieldsOnly>
            <platformMsgs:pageSize>100</platformMsgs:pageSize>
        </platformMsgs:searchPreferences>
        <platformMsgs:passport>
            <platformCore:email>jdoe@netsuite.com</platformCore:email>
            <platformCore:password>mypassword</platformCore:password>
            <platformCore:account>000034</platformCore:account>
            <platformCore:role internalId="3"/>
        </platformMsgs:passport>
    </soapenv:Header>
    <soapenv:Body>
        <platformMsgs:searchMoreWithId>
            <searchId>WEBSERVICES_528736_07012008543995006307049233_d53ef4d2273b15</searchId>
            <pageIndex>2</pageIndex>
        </platformMsgs:searchMoreWithId>
    </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response for Getting the Next Page Using searchMoreWithId

```
<soapenv:Envelope>
    <soapenv:Header>
        <platformMsgs:documentInfo>
            <platformMsgs:nsId>WEBSERVICES_528736_07012008543995006307049233_d53ef4d2273b15</pl
atformMsgs:nsId>
        </platformMsgs:documentInfo>
    </soapenv:Header>
    <soapenv:Body>
        <platformMsgs:searchMoreWithIdResponse>
            <platformCore:searchResult>
         <platformCore:status isSuccess="true"/>
         <platformCore:totalRecords>231</platformCore:totalRecords>
         <platformCore:pageSize>100</platformCore:pageSize>
         <platformCore:totalPages>3</platformCore:totalPages>
         <platformCore:pageIndex>1</platformCore:pageIndex>
        <platformCore: searchId > WEBSERVICES_528736_07012008543995006307049233_d53ef4d2273b15
             </platformCore: searchId >
                <platformCore:recordList>
                    <platformCore:record internalId="5" externalId="entity-5" xsi:type="listRel
:Contact">
                        <listRel:entityId>mike</listRel:entityId>
                        <listRel:firstName>Mike</listRel:firstName>
                        .....
                    </platformCore:record>
                </platformCore:recordList>
            </platformCore:searchResult>
        </platformMsgs:searchMoreWithIdResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Java

```
public List<Contact> searchContactsByCity(NetSuitePortType port, String city) throws Exception
{
    SearchResult searchResult = null;         // search result for each page
    Record [] contacts = null;                    // contacts for each page
    List<Contact> consolidatedResults = new ArrayList<Contact>();   // to return

    // build search criteria
    ContactSearchBasic contactSearch = new ContactSearchBasic();
    contactSearch.setCity(new SearchStringField(city, SearchStringFieldOperator.is));

    // execute the search
    searchResult = port.search(contactSearch);

    if( searchResult.getTotalRecords() > 0 )
    {
        // retain the search ID  to get more pages
        String sSearchId = searchResult.getSearchId();

        // process first page
        contacts = searchResult.getRecordList().getRecord();
        for (Record contact : contacts)
            consolidatedResults.add((Contact)contact);

        // process remaining pages using search ID
        int iNumPages = searchResult.getTotalPages();
        for ( int i=2; i<=iNumPages; i++)
        {
            // get the page
            searchResult = port.searchMoreWithId(sSearchId, i);

            // process the page
            contacts = searchResult.getRecordList().getRecord();
            for (Record contact : contacts)
                consolidatedResults.add((Contact)contact);
        }
    }
    return consolidatedResults;
}
```

searchNext

The searchNext operation is used to retrieve the next set of records after an initial search operation.

Important: Users who authenticate to NetSuite through login can use either searchNext or searchMoreWithId to paginate through search results. Users who authenticate to NetSuite by providing user credentials in the header of their SOAP requests can use **only** searchMoreWithId to paginate through search results, since searchNext requires an active session.

Note: (See Request-Level Credentials for information on request-level-credential authentication.)

Request

The SearchNextRequest type is used for the request. It does not contain any fields.

Response

The SearchNextResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
status	Status	The status for this search. All applicable errors or warnings will be listed within this type.
totalRecords	xsd:int	The total number of records for this search. Depending on the pageSize value, some or all the records may be returned in this response
pageSize	xsd:int	The page size for this search.
totalPages	xsd:int	The total number of pages that are part of this search.
pageIndex	xsd:int	The page index for the current set of results.
searchId	string	Returns a specific search based on its search ID.
recordList	Record[]	A list of records that meet the criteria for this search. The actual records returned need to be of a type that extends the abstract type of record.
searchRowList	SearchRowList	A list of return columns that meet the criteria for this search.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidCredentialsFault
- InvalidSessionFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request

```
<soap:Body>
   <platformMsgs:searchNext/>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
<searchNextResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<searchResult xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
<status isSuccess="true"/>
<totalRecords>93</totalRecords>
<pageSize>10</pageSize>
<totalPages>10</totalPages>
<pageIndex>3</pageIndex>
<recordList>
   <record internalId="865" xsi:type="ns1:Customer"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ns1="urn:relationships_2017_1.lists.webservices.netsuite.com">
      <ns1:entityId>John Bauer</ns1:entityId>
      <ns1:isInactive>false</ns1:isInactive>
      .
      ...[more fields]
      .
      <ns1:customFieldList>
         <customField internalId="custentity_map" xsi:type="StringCustomFieldRef">
            <value>http://maps.google.com</value>
         </customField>
      </ns1:customFieldList>
   </record>
   .
   ...[more records]
   .
</recordList>
```

```
</searchResult>
</searchNextResponse>
</soapenv:Body>
```

ssoLogin

The ssoLogin operation provides a mechanism for a partner application to log in to NetSuite on behalf of the user. This approach enables login to take place without the user's credentials ever going through the partner servers. To use this operation, the inbound single sign-on feature must be enabled.

For inbound single sign-on to work, a single sign-on mapping must exist for every user that accesses NetSuite through the external application. This mapping can be created manually when the user logs in, or by using the mapSso operation.

The following steps outline the general workflow that occurs the first time the user logs in, if the user is creating the mapping manually.

1. User authenticates at the partner site.
2. Partner provides the single sign-on link to user.
3. Partner produces a token. The token is generated from a string that includes three pieces of data: a timestamp, the remote company ID from the external system, and the remote user ID from the external system. This string is encrypted by using a private key.
4. The token enables the user to be redirected to NetSuite.
5. The user logs in to NetSuite, and the authentication link for the user is created.
6. After the single sign-on mapping is created, the partner invokes ssoLogin operation via web services to access NetSuite on behalf of the user.

For more details about the inbound SSO process, see the help topic Inbound Single Sign-on.

Please note the following important details about the ssoLogin operation:

- SsoPassports cannot be handled at the request level.
- If you want to use inbound single sign-on, you must purchase the “Single Sign On Configuration” line item. Contact your account manager for assistance.
- The SuiteTalk mapSso operation is not directly connected to the ssoLogin operation. The purpose of mapSso is to establish the mapping between a user's identity in an external application and the user's identity in NetSuite-this can be done instead of the user logging in to NetSuite on the first single sign-on access.
- The ssoLogin operation is for establishing **inbound** connections only. To create an outbound connection that goes from NetSuite to a third-party application, see the help topic SuiteSignOn Overview.

Important: NetSuite hosts customer accounts in multiple data centers. For that reason, the correct URL for web services access varies depending on the data center hosting the account. Your integration **must** incorporate logic that dynamically determines the correct URL. With the 2012.2 and later endpoints, you should use the getDataCenterUrls operation to dynamically discover the correct URL. With older endpoints, you should use the REST roles service. For details, see the help topic Using the REST roles Service to Get User Accounts, Roles, and Domains.

Request

The SsoPassportRequest type is used for this request. It contains the following fields.

Element Name	XSD Type
SsoPassport	SsoPassport

The following table describes the fields in the SsoPassport complex type. All are required.

Field	Notes
authenticationToken	The authenticationToken is generated by using three values: a timestamp, the partnerAccount value, and the partnerUserId value. For more information about generating a token, see Example Single Sign-on Token and Redirect URLs.
partnerId	This value identifies the partner, or external system. NetSuite uses this data to retrieve the decryption key needed to process the token.
partnerAccount	This value is sometimes referred to as the remote company ID. It is used to identify the user in the external system. **Note:** Starting with the 2015.2 endpoint, this parameter is required.
partnerUserId	This value is sometimes referred to as the remote user ID. It is used to identify the user in the external system. **Note:** Starting with the 2015.2 endpoint, this parameter is required.

Response

The SsoLoginResponse type is used for the response.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
wsRoleList	WsRoleList	Returns a list of roles available for the specified user.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidAccountFault
- InvalidCredentialsFault
- InvalidVersionFault
- ExceededRequestLimitFault
- UnexpectedErrorFault

Sample Code

The following snippets show how to use the ssoLogin operation.

SOAP Request

```
      <urn:ssoLogin xmlns:urn="urn:messages_2017_1.platform.webservices.netsuite.com" xmlns:urn
1="urn:core_2017_1.platform.webservices.netsuite.com">
         <urn:ssoPassport>
           <urn1:authenticationToken>57E1A7DA1CD637E6BD1F6D7AF3F9EE96B0DE6D1E0D337678606BE44487
60CEC5D249031CA0B4618EB50C731703DDE27150F663C7AC11D3B4CEB84329DB2EBE58FE1D16520FF3271476F069C31
DD0BCDA0AAFB4BEF1C3EC0F7DD98579D7314F6A0AD5B3D22AD1A2E766E471B0FA22B1BFEDE4Br58Frr315EC3C7BC1C6
5CFA9A37</urn1:authenticationToken>
           <urn1:partnerId>1</urn1:partnerId>
           <urn1:partnerAccount>35185</urn1:partnerAccount>
           <urn1:partnerUserId>john.smith</urn1:partnerUserId>
         </urn:ssoPassport>
      </urn:ssoLogin>
```

Java

```
public void ssoLogin() throws Exception
{

   String myPartnerAccount= "35185";
   String myPartnerUserId = "john.smith";

   SsoPassport sso = new SsoPassport();
   sso.setPartnerId("1");
   sso.setPartnerAccount(myPartnerAccount);
   sso.setPartnerUserId(myPartnerUserId);

   sso.SetAuthenticationToken(generateToken(myPartnerAccount, myPartnerUserId));

   sessMgr.getPort().ssoLogin(sso);
}
```

update

The update operation is used to update an instance of a record in NetSuite. It is similar to the updateList operation, which allows users to update more than one record at a time.

Only the fields that have been populated in each submitted record are updated in the system. If a field has NOT been populated, it is not updated in the system and it retains its previous value. If a field is set to an empty string, the previous value of the field is replaced with an empty string. Therefore, when updating records, it is recommended that you get the desired record, instantiate a new record of the same type, populate only the fields that require an update and then submit the updated record. This ensures that only the fields requiring an update are written on submission.

Important: Calculated and hidden fields in records are always updated by the system unless your service explicitly overrides the system values. For more information, see Hidden Fields. Also, custom fields can only be set to NULL by submitting the field in nullFieldList. For more information, see CustomFieldList.

To ensure that the most recent data for a specific record is being modified, when a Web service request is received, the values for that record are retrieved at the time of the Update request rather than with the initial Get of the associated record. The record is then updated by the values submitted in the request. It is possible that between the time of the retrieval of the record field values and the submission of the updated fields that the record is altered from another source (for example from a UI submission). In this case an error message is returned to indicate that the fields have been modified since your service retrieved the record.

Although records of a particular type may be used in multiple integration scenarios, each record instance can only have a single external ID value. To maintain data integrity, only a single integrated application can set and update external ID values for each record type. External ID values for all records of a particular type must all come from the same external application.

Note: As of the 2011.2 endpoint, when a record is updated and the values that are sent are exactly the same as the existing record values, the record is not reset (nothing happens). In previous endpoints, sometimes records were reset in these cases.

Request

The UpdateRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
record	Record	Contains an array of record objects. The record type is an abstract type so an instance of a type that extends record must be used—such as Customer or Event.

Response

The UpdateResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
response	WriteResponse	Contains details on the status of the operation and a reference to the updated record.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In the following example, a customer's companyName is updated. The internal ID for the customer must be provided in the request.

```
<soap:Body>
<platformMsgs:update>
   <platformMsgs:record internalId="980" xsi:type="listRel:Customer">
      <listRel:companyName>Shutter Fly Corporation</listRel:companyName>
   </platformMsgs:record>
</platformMsgs:update>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
<updateResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<writeResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"/>

      <baseRef internalId="980" type="customer" xsi:type="ns2:RecordRef"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
</writeResponse>
</updateResponse>
</soapenv:Body>
```

C#

```
 private void updateCustomer()
{
   // This operation requires a valid session
   this.login( true );

   Customer customer = new Customer();

   // Get nsKey for update
   _out.write( "\nEnter nsKey for customer record to be updated : " );
   customer.internalId = _out.readLn().ToUpper();

   // Set name and email
   customer.entityId = "XYZ 2 Inc";
   customer.companyName = "XYZ 2, Inc.";
   customer.email = "bsanders@xyz.com";

   // Populate the address. Updating a list through WS results in the
   // entire contents of the previous list being replaced by the new
   // list.
   CustomerAddressbook address = new CustomerAddressbook();
```

```
    address.defaultBilling = true;
    address.defaultBillingSpecified = true;
    address.defaultBilling = false;
    address.defaultBillingSpecified = true;
    address.label = "Billing Address";
    address.addr1 = "4765 Sunset Blvd";
    address.city = "San Mateo";
    address.state = "CA";
    address.country = Country._unitedStates;

    // Attach the address to the customer
    CustomerAddressbookList addressList = new CustomerAddressbookList();
    CustomerAddressbook[] addresses = new CustomerAddressbook[1];
    addresses[0] = address;
    addressList.addressbook = addresses;
    customer.addressbookList = addressList;

    // Invoke add() operation
    WriteResponse response = _service.update( customer );

    // Process the response
    if ( response.status.isSuccess )
    {
        _out.info(
        "\nThe following customer was updated successfully:" +
        "\nkey=" + ((RecordRef) response.baseRef).internalId +
        "\nentityId=" + customer.entityId +
        "\ncompanyName=" + customer.companyName +
        "\nemail=" + customer.email +
        "\naddressbookList[0].label=" + customer.addressbookList.addressbook[0].label );
    }
    else
    {
        _out.error( getStatusDetails( response.status ) );
    }
}
```

Java

```
public void updateCustomer() throws RemoteException,
ExceededUsageLimitFault, UnexpectedErrorFault, InvalidSessionFault,
ExceededRecordCountFault {
    // This operation requires a valid session
    this.login(true);

    Customer customer = new Customer();

    // Get nsKey for update
    _console.write("\nEnter nsKey for customer record to be updated : ");
    customer.setInternalId(_console.readLn().toUpperCase());

    // Set name and email
    customer.setEntityId("XYZ 2 Inc");
    customer.setCompanyName("XYZ 2, Inc.");
    customer.setEmail("bsanders@xyz.com");

    // Populate the address. Updating a list through WS results in the
    // entire contents of the previous list being replaced by the new
    // list.
    CustomerAddressbook address = new CustomerAddressbook();
    address.setDefaultBilling(Boolean.TRUE);
    address.setDefaultShipping(Boolean.FALSE);
    address.setLabel("Billing Address");
    address.setAddr1("4765 Sunset Blvd");
    address.setCity("San Mateo");
    address.setState("CA");
    address.setCountry(Country._unitedStates);

    // Attach the address to the customer
    CustomerAddressbookList addressList = new CustomerAddressbookList();
    CustomerAddressbook[] addresses = new CustomerAddressbook[1];
    addresses[0] = address;
    addressList.setAddressbook(addresses);
    customer.setAddressbookList(addressList);

    // Invoke add() operation
    WriteResponse response = _port.update(customer);

    // Process the response
    if (response.getStatus().isIsSuccess()) {
       _console.info("\nThe following customer was updated successfully:"
       + "\nkey="
       + ((RecordRef) response.getBaseRef()).getInternalId()
       + "\nentityId="
       + customer.getEntityId()
       + "\ncompanyName="
       + customer.getCompanyName()
       + "\nemail="
       + customer.getEmail()
       + "\naddressbookList[0].label="
       + customer.getAddressbookList().getAddressbook(0)
       .getLabel());
    } else {
       _console.error(getStatusDetails(response.getStatus()));
    }
 }
```

Updating Record Lists

When updating a list of records (a sublist) within a business record you CANNOT update a specific item in the list. Instead you must interact with the sublist as a whole. For details, see Sublists in Web Services.

updateList

The updateList operation is used to update one or more instances of a record type in NetSuite.

If there are multiple records, they can either be of the same record type or different record types. For example, it's possible to update a customer and a contact within a single request using this operation.

Only the fields that have been populated in each submitted record are updated in the system. If a field has not been populated, it is not updated in the system and it retains its previous value. If a field is set to an empty string, the previous value of the field is replaced with an empty string.

Although records of a particular type may be used in multiple integration scenarios, each record instance can only have a single external ID value. To maintain data integrity, only a single integrated application can set and update external ID values for each record type. External ID values for all records of a particular type must all come from the same external application.

Important: Calculated and hidden fields in records are always updated by the system unless your service explicitly overrides the system values. For more information, refer to Hidden Fields.

Note: As of the 2011.2 endpoint, when a record is updated and the values that are sent are exactly the same as the existing record values, the record is not reset (nothing happens). In previous endpoints, sometimes records were reset in these cases.

Request

The UpdateListRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
record[]	Record	Contains an array of record objects. The record type is an abstract type so an instance of a type that extends record must be used—such as Customer or Event.

Note: An asynchronous equivalent is available for this operation, **asyncUpdateList**. For information about asynchronous request processing, see Synchronous Versus Asynchronous Request Processing.

Response

The UpdateListResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
response[]	WriteResponse	Contains an array of WriteResponse objects, each of which contains details on the status of that update operation and a reference to the created record.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault

- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In the following example, two customer records are updated. The first has the single field companyName updated, whereas the second updates two fields: entityID and companyName. The internalID for each record must be provided and the type of record (Customer) to be updated.

```
<soap:Body>
<platformMsgs:updateList>
   <platformMsgs:record internalId="980" xsi:type="listRel:Customer">
      <listRel:companyName>Shutter Fly Corporation</listRel:companyName>
   </platformMsgs:record>
   <platformMsgs:record internalId="981" xsi:type="listRel:Customer">
      <listRel:entityId>GNCC</listRel:entityId>
      <listRel:companyName>GNCC Corp</listRel:companyName>
   </platformMsgs:record>
</platformMsgs:updateList>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
<updateListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<writeResponseList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <writeResponse>
      <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com
"/>
      <baseRef internalId="980" type="customer" xsi:type="ns2:RecordRef"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
   </writeResponse>
   <writeResponse>
      <ns3:status isSuccess="true" xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com
"/>
      <baseRef internalId="981" type="customer" xsi:type="ns4:RecordRef"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com"/>
   </writeResponse>
</writeResponseList>
</updateListResponse>
```

```
</soapenv:Body>
```

C#

```
private void updateCustomerList()
{
   // This operation requires a valid session
   this.login( true );

   // Prompt for list of nsKeys and put in an array
   _out.write( "\nEnter nsKeys for customer records to be updated (separated by commas): " );
   String reqKeys = _out.readLn();
   string [] nsKeys = reqKeys.Split( new Char[] {','} );

   // Create an array of Record objects to hold the customers
   Record[] records = new Record[nsKeys.Length];

   // For each submitted nsKey, populate a customer object
   for ( int i=0; i<nsKeys.Length; i++)
   {
      Customer customer = new Customer();

      // Update name
      customer.entityId = "XYZ Inc " + i;
      customer.companyName = "XYZ, Inc. " + i;

      customer.internalId = nsKeys[i].Trim();
      records[i] = customer;
   }

   // Invoke updateList() operation to update customers
   WriteResponse[] responses = _service.updateList( records );

   // Process responses for all successful updates
   _out.info( "\nThe following customers were updated successfully:" );
   bool hasFailures = false;
   for ( int i=0; i<responses.Length; i++ )
   {
      if ( (responses[i] != null) && (responses[i].status.isSuccess) )
      {
         _out.info( "\nCustomer[" + i + "]:" );
         _out.info(
         "key=" + ((RecordRef) responses[i].baseRef).internalId +
         "\nentityId=" + ((Customer) records[i]).entityId +
         "\ncompanyName=" + ((Customer) records[i]).companyName );
      }
      else
      {
         hasFailures = true;
      }
   }

   // Process responses for all unsuccessful updates
   if ( hasFailures )
   {
      _out.info( "\nThe following customers were not updated:\n" );
      for ( int i=0; i<responses.Length; i++ )
      {
         if ( (responses[i] != null) && (!responses[i].status.isSuccess) )
         {
            _out.info( "Customer[" + i + "]:" );
            _out.info( "key=" + ((RecordRef) responses[i].baseRef).internalId );
            _out.errorForRecord( getStatusDetails( responses[i].status ) );
         }
      }
   }
}
```

Java

```
public void updateCustomerList() throws RemoteException,
ExceededUsageLimitFault, UnexpectedErrorFault, InvalidSessionFault,
ExceededRecordCountFault {
   // This operation requires a valid session
   this.login(true);

   // Prompt for list of nsKeys and put in an array
   _console
   .write("\nEnter nsKeys for customer records to be updated (separated by commas): ");
   String reqKeys = _console.readLn();
   String[] nsKeys = reqKeys.split(",");

   // Create an array of Record objects to hold the customers
   Record[] records = new Record[nsKeys.length];

   // For each submitted nsKey, populate a customer object
   for (int i = 0; i < nsKeys.length; i++) {
      Customer customer = new Customer();

      // Update name
      customer.setEntityId("XYZ Inc " + i);
      customer.setCompanyName("XYZ, Inc. " + i);

      customer.setInternalId(nsKeys[i].trim());
      records[i] = customer;
   }

   // Invoke updateList() operation to update customers
   WriteResponseList responseList = _port.updateList(records);

   // Process responses for all successful updates
   WriteResponse[] responses = responseList.getWriteResponse();
   boolean hasFailures = false;
   _console.info("\nThe following customers were updated successfully:");
   for (int i = 0; i < responses.length; i++) {
      if ((responses[i] != null)
      && (responses[i].getStatus().isIsSuccess())) {
         _console.info("\nCustomer[" + i + "]:");
```

```
            _console.info("key="
            + ((RecordRef) responses[i].getBaseRef()).getInternalId()
            + "\nentityId="
            + ((Customer) records[i]).getEntityId()
            + "\ncompanyName="
            + ((Customer) records[i]).getCompanyName());
         } else {
            hasFailures = true;
         }
      }

      // Process responses for all unsuccessful updates
      if (hasFailures) {
         _console.info("\nThe following customers were not updated:\n");
         for (int i = 0; i < responses.length; i++) {
            if ((responses[i] != null)
            && (!responses[i].getStatus().isIsSuccess())) {
               _console.info("Customer[" + i + "]:");
               _console.info("key="
               + ((RecordRef) responses[i].getBaseRef()).getInternalId());
               _console.errorForRecord(getStatusDetails(responses[i]
               .getStatus()));
            }
         }
      }
   }
```

updateInviteeStatus

The updateInviteeStatus operation lets users respond to NetSuite events that have been sent to them. This operation takes both the internal ID of the event as well as a calendar event status as arguments.

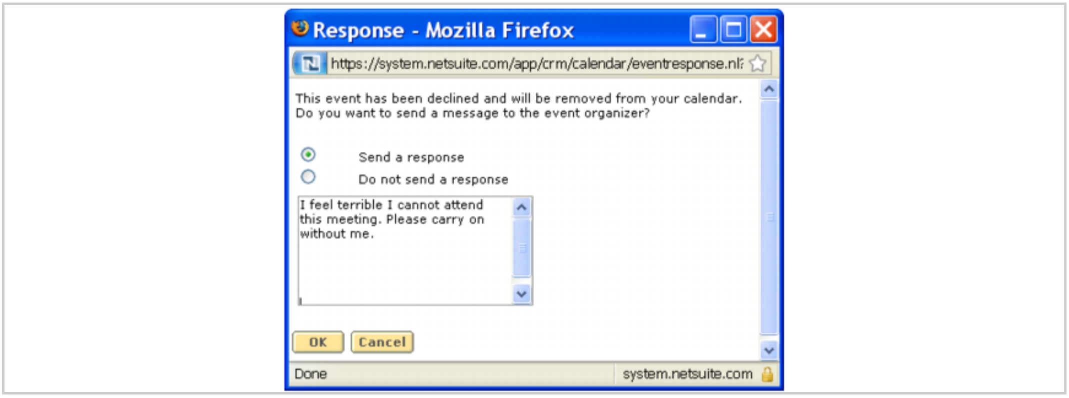
After invitees have responded to the event invitation, the Event record is updated with their response. Possible responses include accepted, declined, tentative, and noResponse.

Note the following:

- For information on the Event record in web services, see the help topic Events (CalendarEvent). For general information on scheduling events in NetSuite, see the help topic Scheduling Events.)
- There is no async equivalent for the updateInviteeStatus operation.

Note the following when using this operation:

- Users must have a valid session. This operation includes a **passport** header to support request level credentials. (For information on request level authentication, see Request-Level Credentials.)
- To update other properties on the Event record, the event owner should still use the update operation. Note, however, event owners are not exempt from using updateInviteeStatus if they have to update their own event response status. There may be cases in which event owners must decline their own event and have another person run the event for them.
- Unlike in the NetSuite UI (see figure), invitees will not be able to send a message back to the organizer. Web services does not currently support messages attached to events. An event invitee's response can include only one of the following values: _accepted, _declined, _tentative, _noResponse.


Response - Mozilla Firefox
https://system.netsuite.com/app/crm/calendar/eventresponse.nl

This event has been declined and will be removed from your calendar.
Do you want to send a message to the event organizer?

Send a response
Do not send a response

I feel terrible I cannot attend this meeting. Please carry on without me.

OK
Cancel
Done
system.netsuite.com

Request

The UpdateInviteeStatusRequest type is used for the request.

Element Name	XSD Type	Notes
eventId	RecordRef	References an existing instance of an Event record.
responseCode	CalendarEventAttendeeRespon se	The CalendarEventAttendeeResponse type includes the following enums: _accepted _declined _tentative _noResponse

Response

The UpdateInviteeStatusResponse type is used for the response.

Element Name	XSD Type	Notes
response	WriteResponse	Contains details on the status of the operation and a reference to the updated record.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

C#

```
NetSuiteService nss = new NetSuiteService();
// login details omitted

UpdateInviteeStatusReference inviteeStatusRef = new UpdateInviteeStatusReference();

// Set the event id for the status update
RecordRef eventIdRef  = new RecordRef();
eventIdRef.internalId = “100”;   // Substitute this with your own event id
inviteeStatusRef.eventId = eventIdRef;

// Set the event status
inviteeStatusRef.responseCode = CalendarEventAttendeeResponse._accepted;

// Update invitee event status
WriteResponse resp = nss.updateInviteeStatus(inviteeStatusRef);
```

SOAP Request

```
<updateInviteeStatus xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <updateInviteeStatusReference>
               <eventId internalId="100" xmlns="urn:core_2017_1.platform.webservices.netsuite.c
om" />
               <responseCode xmlns="urn:core_2017_1.platform.webservices.netsuite.com">_accepte
d
      </responseCode>
            </updateInviteeStatusReference>
         </updateInviteeStatus>
```

SOAP Response

```
  <updateInviteeStatusResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <writeResponse>
               <platformCore:status isSuccess="true"
                  xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
               <baseRef internalId="100" type="calendarEvent" xsi:type="platformCore:RecordRef"

      xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </writeResponse>
         </updateInviteeStatusResponse>
```

updateInviteeStatusList

The updateInviteeStatusList operation is used to update one or more NetSuite events. For general details on the updateInviteeStatus operation, see updateInviteeStatus.

Request

The UpdateInviteeStatusListRequest type is used for the request.

Element Name	XSD Type	Notes
updateInviteeStatusReference[]	UpdateInviteeStatusReference	UpdateInviteeStatusReference uniquely identifies the event and the status you want to update the event with.

The UpdateInviteeStatusReference type includes the following elements:

- eventId
- responseCode

Element Name	XSD Type	Notes
eventId	RecordRef	References an existing instance of an Event record.
responseCode	CalendarEventAttendeeResponse	The CalendarEventAttendeeResponse type includes the following enums: _accepted _declined _tentative _noResponse

Response

The UpdateInviteeStatusListResponse type is used for the response.

Element Name	XSD Type	Notes
response[]	WriteResponse	Contains an array of WriteResponse objects, each of which contains details on the status of that updateInviteeStatusList operation and a reference to each event record.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

C#

```
UpdateInviteeStatusReference [] inviteeStatusRefList = new UpdateInviteeStatusReference[2];
                  for (int i=0; i<2; i++)
                     {
                        RecordRef eventIdRef = new RecordRef();
                         if (i==0)
                           eventIdRef.internalId = “100”;
                         else
                           eventIdRef.internalId = “101”;
                           UpdateInviteeStatusReference statusRef = new UpdateInviteeStatusRefe
rence();
                            statusRef.eventId = eventIdRef;
                            statusRef.responseCode = CalendarEventAttendeeResponse._accepted;
                            inviteeStatusRefList[i] = statusRef;
                     }
                              nss.updateInviteeStatusList(inviteeStatusRefList);
```

SOAP Request

```
<updateInviteeStatusList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <updateInviteeStatusReference>
               <eventId internalId="100" xmlns="urn:core_2017_1.platform.webservices.netsuite.c
om" />
               <responseCode xmlns="urn:core_2017_1.platform.webservices.netsuite.com">_accepte
d
      </responseCode>
            </updateInviteeStatusReference>
            <updateInviteeStatusReference>
               <eventId internalId="101" xmlns="urn:core_2017_1.platform.webservices.netsuite.c
om" />
               <responseCode xmlns="urn:core_2017_1.platform.webservices.netsuite.com">_accepte
d
      </responseCode>
            </updateInviteeStatusReference>
         </updateInviteeStatusList>
```

SOAP Response

```
 <updateInviteeStatusListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com"
>
            <writeResponseList>
               <writeResponse>
                  <platformCore:status isSuccess="true"
          xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
                  <baseRef internalId="100" type="calendarEvent" xsi:type="platformCore:RecordR
ef"
          xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
```

```
                </writeResponse>
                <writeResponse>
                   <platformCore:status isSuccess="true"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
                   <baseRef internalId="101" type="calendarEvent" xsi:type="platformCore:RecordR
ef"
          xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
                </writeResponse>
             </writeResponseList>
          </updateInviteeStatusListResponse>
```

upsert

The upsert operation is used to add a new instance or to update an instance of a record in NetSuite. It is similar to the upsertList operation, which allows users to add or update more than one record at a time.

The upsert operation is similar to both the add and update operations, but upsert can be run without first determining whether a record exists in NetSuite. A record is identified by its external ID and its record type. If a record of the specified type with a matching external ID exists in the system, it is updated. If it does not exist, a new record is created.

Because external ID is mandatory for this operation, upsert is supported only for records that support the external ID field. Also, this operation prohibits the passing of internal ID values.

Note: To prevent duplicate records, NetSuite recommends using external IDs and the upsert and upsertList operations to add records to NetSuite.

Limitations on upsert

- Although records of a particular type may be used in multiple integration scenarios, each record instance can only have a single external ID value. To maintain data integrity, only a single integrated application can set and update external ID values for each record type. External ID values for all records of a particular type must all come from the same external application.
- Upsert cannot complete updates during an initialize / initializeList operation, when a record is transformed into a record of another type. In this case, upsert only adds and does not update records.
- Updates through the upsert operation are also subject to the same limitations as updates through the update operation. For details of these limitations, see update.

Request

The UpsertRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
record	Record	Contains an array of record objects. The record type is an abstract type so an instance of a type that extends record must be used—such as Customer or Event.

Response

The UpsertResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
response	WriteResponse	Contains details on the status of the operation and a reference to thecreated or updated record.

Faults

- This operation can throw one of the following faults. See SOAP Fault Status Codes for more information.
  - InvalidSessionFault
  - InvalidCredentialsFault
  - ExceededRequestLimitFault
  - ExceededUsageLimitFault
  - ExceededRecordCountFault
  - ExceededRequestSizeFault
  - UnexpectedErrorFault
- This operation returns the following run-time error if the passed record type does not support the external ID field:

  ```
  INVALID_RCRD_TYPE:  <record_type> does not support external ID and cannot be used with upsert
  ```

- This operation returns the following run-time error if passed data includes internal ID:

  ```
  USER_ERROR:  You cannot set internalId with upsert.
  ```

- This operation returns the following run-time error if passed data does not include external ID:

  ```
  USER_ERROR:  This operation requies a value for externalId.
  ```

  For more information about error codes, see Error Status Codes.

Sample Code

SOAP Request

```
<soap:Body>
            <upsert xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
               <record xmlns:q1="urn:relationships_2017_1.lists.webservices.netsuite.com" xsi:t
ype="q1:Customer" externalId="THISISMYEXTID">
                  <q1:entityId>XYZ 2 Inc</q1:entityId>
                  <q1:companyName>XYZ 2, Inc.</q1:companyName>
                  <q1:email>bsanders@xyz.com</q1:email>
               </record>
```

```
          </upsert>
       </soap:Body>
```

SOAP Response

```
<soapenv:Body>
            <upsertResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
               <writeResponse>
                  <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.pla
tform.webservices.netsuite.com"/>
                  <baseRef internalId="973" externalId="THISISMYEXTID" type="customer" xsi:type
="platformCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com
"/>
               </writeResponse>
            </upsertResponse>
         </soapenv:Body>
```

C#

```
private void upsertCustomer()
        {
              // This operation requires a valid session
              this.login( true );

              Customer customer = new Customer();

              // Get externalId for upsert
              _out.write( "\nEnter externalId for customer record to be created or updated : "
);
              customer.externalId = _out.readLn().ToUpper();

              // Set name and email
              customer.entityId = "XYZ 2 Inc";
              customer.companyName = "XYZ 2, Inc.";
              customer.email = "bsanders@xyz.com";

              // Invoke upsert() operation
              WriteResponse response = _service.upsert( customer );

              // Process the response
              if (response.status.isSuccess )
              {
                    _out.info(
                    "\nThe upsert operation was successful :" +
                    "\ninternalId=" + ((RecordRef) response.baseRef).internalId +
                    "\nexternalId=" + ((RecordRef) response.baseRef).externalId +
                    "\nentityId=" + customer.entityId +
                    "\ncompanyName=" + customer.companyName);
              } else {
                    _out.error( getStatusDetails( response.status ) );
              }
        }
```

Java

```
public void upsertCustomer() throws RemoteException,ExceededUsageLimitFault,
            UnexpectedErrorFault, InvalidSessionFault,ExceededRecordCountFault
      {
            // This operation requires a valid session
            this.login(true);
            Customer customer = new Customer();

            // Get externalId for add or update
            _console.write("\nEnter externalId for customer record to be updated : ");
            customer.setExternalId(_console.readLn());

            // Set name and email
            customer.setEntityId("XYZ 2 Inc");
            customer.setCompanyName("XYZ 2, Inc.");
            customer.setEmail("bsanders@xyz.com");

            // Invoke upsert() operation
            WriteResponse response = _port.upsert(customer);

            // Process the response
            if (response.getStatus().isIsSuccess())
            {
                  _console.info("\nThe following customer was created/updated successfully:"
                        + "\nkey=" + ((RecordRef) response.getBaseRef()).getInternalId()
                        + "\nexternalId=" + ((RecordRef) response.getBaseRef()).getExternalId()

                        + "\nentityId=" + customer.getEntityId()
                        + "\ncompanyName=" + customer.getCompanyName()
                        + "\nemail=" + customer.getEmail());
            }
            else
            {
                  _console.error(getStatusDetails(response.getStatus()));
            }
      }
```

upsertList

The upsertList operation is used to add or update one or more instances of a record type in NetSuite.

If there are multiple records, they can either be of the same record type or different record types. For example, it's possible to add or update a customer and a contact within a single request using this operation.

The upsertList operation is similar to both the addList and updateList operations, but upsert can be run without first determining whether records exist in NetSuite. Records are identified by their external ID and their record type. If a record of the specified type with a matching external ID exists in the system, it is updated. If it does not exist, a new record is created.

Because external ID is mandatory for this operation, upsertList is supported only for records that support the external ID field. Also, this operation prohibits the passing of internal ID values.

Note: To prevent duplicate records, NetSuite recommends using external IDs and the upsert and upsertList operations to add records to NetSuite.

Limitations on upsertList

- Although records of a particular type may be used in multiple integration scenarios, each record instance can only have a single external ID value. To maintain data integrity, only a single integrated application can set and update external ID values for each record type. External ID values for all records of a particular type must all come from the same external application.
- UpsertList cannot complete updates during an initialize / initializeList operation, when a record is transformed into a record of another type. In this case, upsertList only adds and does not update records.
- Updates through the upsertList operation are also subject to the same limitations as updates through the updateList operation. For details of these limitations, see updateList.

Request

The UpsertListRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
record[]	Record	Contains an array of record objects. The record type is an abstract type so an instance of a type that extends record must be used—such as Customer or Event.

Note: An asynchronous equivalent is available for this operation, **asyncUpsertList**. For information about asynchronous request processing, see Synchronous Versus Asynchronous Request Processing.

Response

The UpsertListResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
response[]	WriteResponse	Contains an array of WriteResponse objects, each of which contains details on the status of that upsert operation and a reference to the created or updated record.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault

- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

This operation returns the following run-time error if the passed record type does not support the external ID field:

```
INVALID_RCRD_TYPE:  <record_type> does not support external ID and cannot be used with upsert
```

This operation returns the following run-time error if passed data includes internal ID:

```
USER_ERROR:  You cannot set internalId with upsert.
```

This operation returns the following run-time error if passed data does not include external ID:

```
USER_ERROR:  This operation requies a value for externalId.
```

For more information about error codes, see Error Status Codes.

Sample Code

SOAP Request

```
<soap:Body>
         <upsertList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record xmlns:q1="urn:relationships_2017_1.lists.webservices.netsuite.com" xsi:type
="q1:Customer" externalId="ext1">
               <q1:entityId>XYZ Inc 0</q1:entityId>
               <q1:companyName>XYZ, Inc. 0</q1:companyName>
            </record>
            <record xmlns:q2="urn:relationships_2017_1.lists.webservices.netsuite.com" xsi:type
="q2:Customer" externalId="ext2">
               <q2:entityId>XYZ Inc 1</q2:entityId>
               <q2:companyName>XYZ, Inc. 1</q2:companyName>
            </record>
         </upsertList>
      </soap:Body>
```

SOAP Response

```
<soapenv:Body>
         <upsertListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <writeResponseList>
               <writeResponse>
                  <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.pla
tform.webservices.netsuite.com"/>
                  <baseRef internalId="970" externalId="ext1" type="customer" xsi:type="platfor
mCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
```

```
                </writeResponse>
                <writeResponse>
                   <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.pla
tform.webservices.netsuite.com"/>
                   <baseRef internalId="974" externalId="ext2" type="customer" xsi:type="platfor
mCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
                </writeResponse>
             </writeResponseList>
          </upsertListResponse>
       </soapenv:Body>
```

C#

```
private void upsertCustomerList()
       {
             // This operation requires a valid session
             this.login( true );

             // Prompt for list of externalIds and put in an array
             _out.write( "\nEnter externalIds for customer records to be updated (separated by
commas): " );
             String reqKeys = _out.readLn();
             string [] nsKeys = reqKeys.Split( new Char[] {’,’} );

             // Create an array of Record objects to hold the customers
             Record[] records = new Record[nsKeys.Length];

             // For each submitted nsKey, populate a customer object
             for ( int i=0; i<nsKeys.Length; i++)
             {
                   Customer customer = new Customer();

                   // Update name
                   customer.entityId = "XYZ Inc " + i;
                   customer.companyName = "XYZ, Inc. " + i;

                   customer.externalId = nsKeys[i].Trim();
                   records[i] = customer;
             }

             // Invoke upsertList() operation to create or update customers
             WriteResponse[] responses = _service.upsertList( records );

             // Process responses for all successful updates
             _out.info( "\nThe following customers were updated or created successfully:" );

             bool hasFailures = false;
             for ( int i=0; i<responses.Length; i++ )
             {
                   if ( (responses[i] != null) && (responses[i].status.isSuccess) )
                   {
                         _out.info( "\nCustomer[" + i + "]:");
                         _out.info( "internalId=" + ((RecordRef) responses[i].baseRef).interna
lId + " externalId="+((RecordRef) responses[i].baseRef).externalId +
                         "\nentityId=" + ((Customer) records[i]).entityId +
                         "\ncompanyName=" + ((Customer) records[i]).companyName );
                   }
                   else
                   {
                         hasFailures = true;
                   }
             }

             // Process responses for all unsuccessful updates
             if ( hasFailures )
             {
                   _out.info( "\nThe following customers were not updated:\n" );
                   for ( int i=0; i<responses.Length; i++ )
                   {
                         if ( (responses[i] != null) && (!responses[i].status.isSuccess) )
                         {
                               _out.info( "Customer[" + i + "]:" );
                               _out.info( "key=" + ((RecordRef) responses[i].baseRef).internal
 Id );
                              _out.errorForRecord( getStatusDetails(responses[i].status ) );
                         }
                   }
             }
       }
```

Java

```
public void upsertCustomerList() throws RemoteException,ExceededUsageLimitFault,
                        UnexpectedErrorFault,InvalidSessionFault,ExceededRecordCountFault
            {
                        // This operation requires a valid session
                        this.login(true);

                        // Prompt for list of externalIds and put in an array
                        _console.write("\nEnter externalIds for customer records to be updated
(separated by commas): ");
                        String reqKeys = _console.readLn();
                        String[] nsKeys = reqKeys.split(",");

                        // Create an array of Record objects to hold the customers
                        Record[] records = new Record[nsKeys.length];

                        // For each submitted nsKey, populate a customer object
                        for (int i = 0; i < nsKeys.length; i++)
                        {
                                    Customer customer = new Customer();

                                    // Update name
                                    customer.setEntityId("XYZ Inc " + i);
                                    customer.setCompanyName("XYZ, Inc. " + i);
                                    customer.setExternalId(nsKeys[i].trim());

                                    records[i] = customer;
                        }

                        // Invoke upsertList() operation to create or update customers
                        WriteResponseList responseList = _port.upsertList(records);

                        // Process responses for all successful upserts
                        WriteResponse[] responses = responseList.getWriteResponse();

                        boolean hasFailures = false;
                        _console.info("\nThe following customers were processed successfully:")
;

                        for (int i = 0; i < responses.length; i++)
                        {
                                    if ((responses[i] != null) && (responses[i].getStatus().isI
sSuccess()))
                                    {
                                                _console.info("\nCustomer[" + i + "]:");
                                               _console.info("key=" + ((RecordRef) responses[i]
.getBaseRef()).getInternalId()
                                                            + "\nexternalId=" + ((RecordRef) re
sponses[i].getBaseRef()).getExternalId()
                                                            + "\nentityId="           + ((Custo
mer) records[i]).getEntityId()
                                                            + "\ncompanyName=" + ((Customer) re
cords[i]).getCompanyName());
                                    }
                                    else
                                    {
                                                hasFailures = true;
                                    }
                        }

                        // Process responses for all unsuccessful updates
                        if (hasFailures)
                        {
                                    _console.info("\nThe following customers were not updated:\
n");
                                    for (int i = 0; i < responses.length; i++)
                                    {
                                                if ((responses[i] != null) && (!responses[i].ge
tStatus().isIsSuccess()))
                                                {
                                                            _console.info("Customer[" + i + "]:
");
                                                            _console.info("key=" + ((RecordRef)
 responses[i].getBaseRef()).getInternalId());
                                                            _console.errorForRecord(getStatusDe
tails(responses[i].getStatus()));
                                                }
                                    }
                        }

            }
```

Web Services PHP Toolkit

If you are new to PHP and SuiteTalk, it is recommended that you review the following topics:

- PHP Toolkit Overview
- Downloading the PHP Toolkit
- Configuring an Environment for the PHP Toolkit
- Creating a Web Services PHP Project
- Creating and Submitting Records Using the PHP Toolkit
- Logging SOAP Requests and Responses Using the PHP Toolkit

Should you run into issues during your development, see Troubleshooting PHP and Web Services.

Important: Using PHP scripts with SuiteTalk requires PHP version 5.3 or later.

PHP Toolkit Overview

NetSuite provides a library and tools to assist in the development of PHP applications that interface to the SuiteTalk platform.

The NetSuite PHP toolkit for SuiteTalk is a core library containing all classes, objects and methods parsed from the WSDL of a SuiteTalk endpoint. The toolkit also comes with several helper functions that can be used to simplify working with arrays and objects.

Important: Each PHP Toolkit is specific to a SuiteTalk endpoint. You must download the correct version of the toolkit for the endpoint you are using. Also, updating to a new endpoint requires an update of the toolkit. **To use the 2017.2 PHP Toolkit, you must upgrade to the 2017.2 WSDL. The 2017.2 PHP Toolkit does not officially support different WSDL versions.**

Updating Existing PHP Code

NetSuite offered a completely revamped PHP Toolkit as part of Version 2012 Release 2. As of 2012.2, the PHP Toolkit has the ability to generate proxy classes for every new NetSuite WSDL, creating a development environment similar to Axis or .NET.

Additionally, the new toolkit uses a different programming model (the use of proxy classes over pure associative arrays, though the latter is still supported).

If you upgrade to any version of the PHP Toolkit later than 2012.2, existing PHP applications created using previous versions of the toolkit will need to be updated. To understand the nature of the coding changes, see the table below. Also see the sample applications that come with the new toolkit.

	PHP Toolkit 2012.2 and Later	**PHP Toolkit Versions Older than 2012.2**
Record object initialization	Each record type can now be initialized using the same record type name. Example - To initialize a Customer record object: $CustomerRec = new Customer();	Initialize the object using the object nsComplexObject. Example - To initialize a Customer record object: $CustomerRec = new nsComplexOb ject('Customer');

	PHP Toolkit 2012.2 and Later	PHP Toolkit Versions Older than 2012.2
Object type Initialization All object types are listed and explained in Platform Enumerations.	Each complex, search and custom field type is initialized using the same object type name: Example - To initialize a basic search for customers: $selectTypeField = new RecordRef(); To initialize a recordRef object: $selectTypeField = new RecordRef();	These objects are initialized using the object nsComplexObject. Example - To initialize a basic search for customers: $search = new nsComplexObject('CustomerSearchBasic'); To initialize a recordRef object: $selectTypeField = new nsComplexObject ('RecordRef');
Calling the NetSuite SuiteTalk (Web Services) operations	The operations are called from the **service** object instantiated from the NetSuiteService class (NetSuiteService.php). Example - $service = new NetSuiteService(); ... $addResponse = $service->add($request);	The operations are called from the **client** object instantiated from the nsClient class (declared on the phptoolkit.php). Example - $myNSclient = new nsClient(nsHost::live); ... $addResponse = $myNSclient->add($contact);
Request object	You must instantiate a request object depending on the kind of operation. This request object will accept the Record object, which will then be set to the operation. Each operation has a corresponding request type. ■ AddRequest for add operation ■ AddListRequest for addList operation ■ SearchRequest for search operation. Example - To add a customer: $service = new NetSuiteService(); $CustomerRec = new Customer(); ... $request = new AddRequest(); $request->record = $CustomerRec; ... $addResponse = $service->add($request);	None. The record object is directly set to the operation. Example - To add a customer: $CustomerRec = new nsComplexObject('Customer'); ... $addResponse = $myNSclient->add ($CustomerRec);
The setFields method	A function to set an array of field objects to the record object. Example - $customerRec = new Customer(); $customerFields = array ('isPerson' => true, 'firstName' => $firstName, 'lastName' => $lastName, 'companyName' => $companyName);	This is a method of the record object used to set the array of field values to the record. Example - $customerFields = array ('isPerson' => true, 'firstName' => $firstName, 'lastName' => $lastName, 'companyName' => $companyName); // create Customer record

	PHP Toolkit 2012.2 and Later	PHP Toolkit Versions Older than 2012.2
	setFields($customerRec, $ customerFields);	$customer = new nsComplexObject ('Customer'); // set fields $customer->setFields($ customerFields);

Downloading the PHP Toolkit

To download the 2017.2 PHP Toolkit, click the link for the PHPToolkit_2017_2.zip file.

The toolkit is provided as an archive file and contains:

- A folder called PHPToolkit that contains:
  - NetSuiteService.php – Class library
  - NSconfig.php – Passport configuration file
  - NSPHPClient.php – Helper functions
- A folder called Samples that contains PHP code examples that are usable in your own projects.

Next see, Configuring an Environment for the PHP Toolkit.

Configuring an Environment for the PHP Toolkit

You can run PHP scripts from the command line by installing a PHP interpreter. Command line scripts are suitable for stand-alone applications and are useful for scripting and testing. See Installing PHP for the Command Line for more details.

The PHP Toolkit's real power, however, is in the ability to use a web server-based PHP interpreter module to integrate NetSuite into an existing web presence, whether it is an external e-commerce website, or an intranet-only Human Resources portal, for example.

A web server executes a PHP script through the included PHP interpreter module. PHP scripts can be text files containing PHP code, or they can be HTML web pages containing PHP tags indicating sections of PHP code. See Installing a Web Server for PHP for more details.

Note: For information on downloading the PHP Toolkit, see Downloading the PHP Toolkit.

Installing PHP for the Command Line

Many distributions of Linux and Mac OS X® are pre-installed with PHP. Packages for Windows® can be obtained from http://windows.php.net/download/.

After PHP is installed, you need to edit the php.ini file. The PHP Toolkit needs to use certain libraries that are not enabled in a PHP installation by default.

In the php.ini configuration file, in the section [Dynamic Extensions], the following lines need to be added if they do not exist:

```
extension=php_soap.dll
extension=php_openssl.dll
```

The modules indicated by the .dll extensions pertains to a Windows installation. If you are configuring a Unix-based platform, change **.dll** to **.so**:

```
extension=php_soap.so
extension=php_openssl.so
```

Note: For a Windows installation of PHP, change the caching directory to something similar to:

soap.wsdl_cache_dir="c:/Windows/Temp"

Installing a Web Server for PHP

Note: This section talks about using Apache Web Server, but any web server that supports PHP, SSL, and SOAP should be able to be used for SuiteTalk.

The simplest way to provision a web server for the PHP Toolkit is to install a LAMP package. LAMP is the acronym for Linux, Apache, MySQL, and PHP. After a LAMP package is installed, a complete web server with database and PHP scripting functionality is available with very little configuration required. The PHP Toolkit does require, however, that you make a few modifications to the default LAMP installation.

Note: There are several LAMP distributions available free of charge for most computer architectures. A well known package for Windows® is WAMP, which is available at http://www.wampserver.com/en/. Follow the instructions that come with the package to install and initially configure it.

After LAMP is installed, the php.ini and the httpd.conf configuration files need to be edited. Note that the LAMP installation will not use any existing PHP installation; it will install its own self-contained PHP package.

In the php.ini configuration file, in the section [Dynamic Extensions], the following lines need to be added if they do not exist:

```
extension=php_soap.dll
extension=php_openssl.dll
```

The modules indicated by the .dll extensions pertains to a Windows installation. If you are configuring a Unix-based platform, change **.dll** to **.so** :

```
extension=php_soap.so
extension=php_openssl.so
```

The Apache Web Server needs to be told to load the PHP module so that it can run PHP scripts. In the httpd.conf configuration file in the section Dynamic Shared Object Support, the following line must be added. (Note that the line may already exist; in that case check that the path to the PHP module is correct.)

```
LoadModule php5_module  C:\<PATH TO PHP>\modules\libphp5.dll
```

The module indicated by the .dll extension pertains to a windows installation. If you are configuring a Unix-based platform, the PHP module will have a **.so** extension. Note also that the path to the PHP module will differ:

```
LoadModule php5_module  /<PATH TO PHP>/bin/php/modules/libphp5.so
```

Important: As indicated in these examples, the name of the module is libphp5.dll or libphp5.so, but be aware that SuiteTalk requires PHP 5.**3** or later. In other words, the name of the module is not specific to the exact PHP version. (However, the exact version is likely indicated by the <PATH TO PHP> folder name.)

Important: After a change to any configuration files, LAMP's services must be restarted.

Important: If you are running PHP scripts from the command line **and** through a web server, there will be **two** php.ini files that need to be edited, one for the command line PHP installation, and one provided by the LAMP installation.

Note: For a Windows installation of PHP, change the caching directory to something similar to:

soap.wsdl_cache_dir="c:/Windows/Temp"

Creating a Web Services PHP Project

For each project you need to perform the following steps:

1. Copy the PHPToolkit directory to the directory in your IDE where your project resides.
2. Edit the NSconfig.php file, substituting your own account details.
3. If SOAP logging is required, create a folder called **nslog** inside the PHPToolkit folder. See Logging SOAP Requests and Responses Using the PHP Toolkit for more details.
4. Each solution needs to include the NetSuiteService.php file by adding the following line to the main source file:

   ```
   require_once '../PHPToolkit/NetSuiteService.php';
   ```

   This step will prompt most IDEs to parse the NetSuiteService.php file for auto completion information.
5. Instantiate a service:

```
$service = new NetSuiteService();
```

Important: Each time you upgrade your NetSuite endpoint, for example from WSDL version 2017.1 to 2017.2, you will also need upgrade the NetSuiteService.php class library to take advantage of any new features.

In 2016.1, Token-based Authentication support was added to the PHPToolkit. There is a sample token_based_authentication.php that details how to use token-based authentication with PHPToolkit in the PHPToolkit_2017_2.zip Samples folder. The following guidelines should be followed when using token-based authentication with PHPToolkit:

- Each request requires a unique TokenPassport
- NetSuite does not generate TokenPassports. You must provide your own implementation of iTokenPassportGenerator
- You configure NetSuite to use your iTokenPassportGenerator by calling the setTokenGenerator:

```
$service = new NetSuiteService();
$service = setTokenGenerator($generator);
```

For more information about token-based authentication, see the help topic Token-based Authentication and Web Services.

Creating and Submitting Records Using the PHP Toolkit

These sections provide basic examples for creating and submitting NetSuite records using the PHP toolkit.

- Creating a Record with PHP
- Creating a Record with the setFields Method
- Submitting a Record

Creating a Record with PHP

The following example shows how to create and submit a Purchase Order. In this case, the Purchase Order object is created then populated one field at a time.

Note that after the Purchase Order is instantiated, the IDE should automatically start suggesting members of the PurchaseOrder object using its code completion mechanism.

```
// Include the PHP Toolkit Library
require_once '../PHPToolkit/NetSuiteService.php';

// Instantiate a service
$service = new NetSuiteService();

// Create Purchase Order object
$po = new PurchaseOrder();

// Create entity record
$po->entity = new RecordRef();

// PO internal ID = 38
$po->entity->internalId = 38;

// Create PO Item sublist
$po->itemList = new PurchaseOrderItemList();

// Create PO Item
$poi = new PurchaseOrderItem();
$poi->item = new RecordRef();
```

```
// Set Item details
$poi->item->internalId = 15;
$poi->quantity = 5;

// Attach items array to PO
$po->itemList->item = array($poi);
```

Creating a Record with the setFields Method

An alternative method to create the object is to use array notation to set the object fields. The advantage is more compact code; the disadvantage is that code completion will not work.

The following example creates the same Purchase Order object using the setFields method:

```
// Create the Purchase Order object
$po = new PurchaseOrder();

// Create array of fields
$purchaseOrderFields = array('entity' => array('internalId' => 38), 'itemList' => array('item'
=> array( array('item' => array('internalId' => 15), 'quantity' => 5)) ) );

// Call setFields to recursively create objects and populate their fields
setFields($po, $purchaseOrderFields);
```

In addition to creating and populating objects, the setFields method performs basic type and field checking to make sure the array structure corresponds to the expected object hierarchy and types.

Note: The setFields method is defined in the NSPHPClient.php helper file, which is included automatically by NetSuiteService.php

Submitting a Record

After the object is created and populated, the object can be submitted to NetSuite in one of the operations supported by the type of object you have created. The following code sample shows the **add** operation:

```
// Create Add request
$request = new AddRequest();

// Attach the Purchase Order to the request
$request->record = $po;

// Submit the record and save the response
$addResponse = $service->add($request);
```

The **add** operation returns a ReadResponse object that is used to access the result of the operation. The following code sample reads and displays a simple textual indication of the result:

```
if (!$addResponse->writeResponse->status->isSuccess)
{
   echo "ADD ERROR";
}
else
{
   echo "ADD SUCCESS, id " . $addResponse->writeResponse->baseRef->internalId;
}
```

Note: The PHP Toolkit package contains more code examples in the Samples folder.

Logging SOAP Requests and Responses Using the PHP Toolkit

To enable SOAP logging with the PHP Toolkit, create a folder called **nslog** in the **PHPToolkit** folder inside the project. The PHP Toolkit will automatically spool the SOAP requests generated by the PHP code, and the responses that have been returned by the server, into the nslog folder in standard XML format.

The following listing shows the typical project structure when logging is enabled:

```
Project_1 <Directory>
   PHPToolkit.php
   NSconfig.php
   NetSuiteService.php
   nslog <Directory>
      20120317.160138.6941-add-request.xml
      20120317.165167.8345-add-response.xml
      20120317.180653.3467-delete-request.xml
      20120317.187474.3356-delete-response.xml
      ...
```

Note: To disable logging, rename or delete the nslog folder.

Troubleshooting PHP and Web Services

The following table provides solutions for problems that may be encountered when using PHP with NetSuite’s web services.

Problem	Solution
The following error is returned when attempting to send requests over https: Fatal error: Uncaught SoapFault exception: [WSDL] SOAP-ERROR: Parsing WSDL: Couldn't load from 'https://webservices.netsuite.com/wsdl /v2017_2_0/netsuite.wsdl'.	Edit the php.ini file by uncommenting "extension=php_openssl.dll" under Dynamic Extensions. Windows only: Make sure that libeay32.dll and ssleay32.dll files are in your path. You can do this by copying the files to your System32 folder.
The following time out error is returned: Fatal error: Maximum execution time of 30 seconds exceeded.	Edit the php.ini file and change max_execution_time - 30 (or default value) to 200.
The following error message is returned:	Edit the php.ini file and by setting default_socket_timeout = 200.

Problem	Solution
Debug Error: Uncaught SoapFault exception: [HTTP] Error Fetching http headers.	
The following error message is returned: Warning: It is not yet possible to assign complex types to properties.	Open the php.ini file and edit error_reporting. Set it to E_ERROR instead of E_ALL.
The execution of script stops and error was not shown, or notices are being shown on the page.	Open the php.ini file and set error_reporting = E_ERROR & ~E_NOTICE.
The following error is returned: Fatal error: Allowed memory size of 8388608 bytes exhausted (tried to allocate 3024 bytes).	Open the php.ini file and set memory_limit = 80M.
Performance issue: Every time a web services request executes, PHP performs a GET on the wsdl.	For a non-Windows installation of PHP, look for property soap.wsdl_cache_dir in the php.ini file and make sure that the specified folder ("/tmp" -> c:\tmp) exists in the host. **Note:** For a **Windows** installation of PHP, change the caching directory to something similar to: soap.wsdl_cache_dir="c:/Windows/Temp"
You have made a change in your php.ini file, yet the value you set is not respected.	Make sure that the property exists only one time in the php.ini file.

Web Services Error Handling and Error Codes

This section provides information on the following:

- The three exception types that are supported in the SuiteTalk: warnings, errors, and faults. See Web Services Warnings, Errors, and Faults for an overview.
- SOAP Faults that can be thrown for each SuiteTalk operation. See SOAP Faults for Each Operation.
-
    - SOAP Fault Status Codes
    - Error Status Codes
    - Warning Status Codes

Web Services Warnings, Errors, and Faults

Based on the error code or fault that is received, the client can take the appropriate action.

- Warnings: informational notifications requiring an action within the UI but requiring no response from a Web service call. Data may or may not be processed depending on preference settings. For more information on preferences, see Company-Wide Preferences.
- Errors: exceptions returned on a record-by-record basis due to invalid or incomplete data. The service is processed as requested, however, only those records without errors are updated.
- Faults: a fundamental exception type that results in the entire request not being processed.

Warnings

A warning is a notification sent to a user to prevent a subsequent error or to ensure better data quality. In the Netsuite UI, a warning is presented to the user through a dialog box and generally requires an action from the user. Since there is no interaction of this nature in the web services model, the request must specify what to do in the case of a warning. The options are:

- Ignore the warning and submit the record to the database
- Heed the warning and abort the submission — treat as an error

  You can set a company wide preference on the Web Services Preferences page on how to handle warnings or you can specify how warnings should be handled in a specific request. Request level preferences override company-wide preferences. Refer to Company-Wide Preferences for information on how to set company-wide preferences.

  Following is an example of a warning message in the response to a request to add a customer record without a zip code (with the Treat Warnings as Errors preference set to false):

Code

```
<addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <writeResponse>
      <platformCore:status isSuccess="true" xmlns:platformCore="urn:core.platform_2017.1.platfo
rm.webservices.netsuite.com">
         <platformCore:statusDetail type="WARN">
            <platformCore:code>WARNING</platformCore:code>
         <platformCore:message>Without a zip/postal code it will not be possible to use
```

```
          this address with 3rd Party shippers. Click Cancel to edit the address.
         </platformCore:message>
         </platformCore:statusDetail>
      </platformCore:status>
      <baseRef internalId="50" type="customer" xsi:type="platformCore:RecordRef" xmlns:platform
Core="urn:core.platform_2017.1.platform.webservices.netsuite.com"/>
   </writeResponse>
</addResponse>
```

The above customer was added because no error occurred. However, future errors could be avoided by responding appropriately to the warning message.

For a detailed list of all warning messages and the associated codes generated by NetSuite, see Warning Status Codes.

Errors

Errors result if invalid or incomplete data is submitted when performing an operation. For example, if a client attempts to update a customer record with an invalid internal ID, the operation fails and the response contains an error code and message.

When numerous records are submitted within the same request, each is treated individually. For example, if a client attempts to update two events within the same request, where one record has invalid data and the other has valid data, only one of the records has an error and is not updated.

Code

```
<updateListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <writeResponseList>
      <platformCore:status isSuccess="true" xmlns:platformCore="urn:core.platform_2017_1.platfo
rm.webservices.netsuite.com"/>
      <writeResponse>
         <platformCore:status isSuccess="false" xmlns:platformCore="urn:core.platform_2017_1.pl
atform.webservices.netsuite.com">
            <platformCore:statusDetail type="ERROR">
               <platformCore:code>RCRD_DSNT_EXIST</platformCore:code>
               <platformCore:message>That record does not exist. Invalid record:type=event,id=2
22,scompid=3604360</platformCore:message>
            </platformCore:statusDetail>
         </platformCore:status>
         <baseRef internalId="222" type="calendarEvent" xsi:type="platformCore:RecordRef" xmlns
:platformCore="urn:core.platform_2017_1.platform.webservices.netsuite.com"/>
      </writeResonse>
      <writeResponse>
         <platformCore:status isSuccess="true" xmlns:platformCore="urn:core.platform_2017_1.pla
tform.webservices.netsuite.com"/>
         <baseRef internalId="220" type="calendarEvent" xsi:type="platformCore:RecordRef" xmlns
:platformCore="urn:core.platform_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </writeResponseList>
</updateListResponse>
```

For a detailed list of all error messages and the associated codes, see Error Status Codes.

Faults

Faults are exceptions that are of a more fundamental nature than errors. A key distinction between errors and faults is that a fault prevents any operation within a request from being processed whereas an error prevents only a single operation from succeeding on an individual record.

For example, continuing from the case above, if the user's session has timed out when making the request, neither update for either event record is processed and an invalidSessionFault is returned in the response.

Expand code

```
<soapenv:Fault>
   <faultcode>soapenv:Server.userException</faultcode>
   <faultstring>Your connection has timed out. Please log in again.
   </faultstring>
   <detail>
      <platformFaults:invalidSessionFault xmlns="urn:faults.platform_2017_1.platform.webservice
s.netsuite.com">
         <platformFaults:code>INVALID_SESSION</platformFaults:code>
         <platformFaults:message>Your connection has timed out. Please log in again.</platformF
aults:message>
      </platformFaults:invalidSessionFault>
      <ns1:hostname xmlns:ns1="http://xml.apache.org/axis">partners-java10009.bos.netledger.com

      </ns1:hostname>
   </detail>
</soapenv:Fault>
```

SOAP uses the detail element to capture the error code through the code element and the error message through the message element. The faultcode and faultstring are automatically populated by the server.

Following is an example of a SOAP fault named InvalidCredentialsFault for an invalid e-mail address that is returned as part of the login operation.

Code

```
<soapenv:Fault>
      <faultcode>soapenv:Server.userExveption</faultcode>
      <faultstring>You have entered an invalid e-mail address or account number. Please try aga
in.</faultstring>
      <detail>
         <platformFaults:invalidCredentialsFault xmlns=”urn:faults.platform_2017_1.platform.web
services.netsuite.com”>
            <platformFaults:code>INVALID_LOGIN_CREDENTIALS</platformFaults:code>
            <platformFaults:message>You have entered an invalid e-mail address or account numbe
r. Please try again.</platformFaults:message>
         </platformFaults:invalidCredentialsFault>
         <ns1:hostname xmlns:ns1="http://xml.apache.org/axis">partners-java10009.bos.netledger.
com
```

```
        </ns1:hostname>
    </detail>
</soapenv:Fault>
```

For a detailed list of fault messages and the associated codes, see SOAP Fault Status Codes. The list does not contain every SOAP fault status code. Some SOAP faults do not have a special fault code.

Important: Unexpected errors return an error ID that can be provided to NetSuite Support to help them isolate the error in your account. The following is an example of the error ID:

```
    <platformCore:code>UNEXPECTED_ERROR</platformCore:code>
    <platformCore:message>An unexpected error occurred. Error ID: fevsjhv41tji2juy3le73</plat
formCore:message>
```

Error IDs will be especially helpful for async operations, as the ID will indicate the error that occurred during the execution of the job, not the retrieval. The exception is if the async error occurred at the platform-level. In this case, the stack will be attached to the job itself.

SOAP Faults for Each Operation

The following table lists the SOAP fault types that can be thrown for each SuiteTalk operation.

Login throws...
ACCT_TEMP_UNAVAILABLE
EMAIL_ADDRS_REQD
INVALID_ACCT
INVALID_LOGIN_CREDENTIALS
INVALID_VERSION
PSWD_REQD
UNEXPECTED_ERROR
WS_CONCUR_SESSION_DISALLWD
WS_FEATURE_REQD
WS_PERMISSION_REQD

MapSso throws...
ACCT_TEMP_UNAVAILABLE
EMAIL_ADDRS_REQD
INVALID_ACCT
INVALID_LOGIN_CREDENTIALS
INVALID_VERSION
PSWD_REQD
UNEXPECTED_ERROR

MapSso throws...
WS_CONCUR_SESSION_DISALLWD
WS_FEATURE_REQD
WS_PERMISSION_REQD

Logout throws...
INVALID_VERSION
UNEXPECTED_ERROR
WS_CONCUR_SESSION_DISALLWD
SESSION_TIMED_OUT
WS_REQUEST_BLOCKED

add / update / delete throw...
INVALID_VERSION
SESSION_TIMED_OUT
UNEXPECTED_ERROR
USER_ERROR
WS_CONCUR_SESSION_DISALLWD
WS_LOG_IN_REQD

addList / updateList / deleteList throw...
INVALID_VERSION
MAX_RCRDS_EXCEEDED
SESSION_TIMED_OUT
UNEXPECTED_ERROR
USER_ERROR
WS_CONCUR_SESSION_DISALLWD
WS_LOG_IN_REQD

getList/getAll throw...
INVALID_VERSION
MAX_RCRDS_EXCEEDED
SESSION_TIMED_OUT
UNEXPECTED_ERROR
USER_ERROR
WS_CONCUR_SESSION_DISALLWD
WS_LOG_IN_REQD

get throws...
INVALID_VERSION
SESSION_TIMED_OUT
UNEXPECTED_ERROR
USER_ERROR
WS_CONCUR_SESSION_DISALLWD
WS_LOG_IN_REQD

search / searchNext / searchMore throw...
INVALID_VERSION
MAX_RCRDS_EXCEEDED
SESSION_TIMED_OUT
UNEXPECTED_ERROR
USER_ERROR
WS_CONCUR_SESSION_DISALLWD
WS_LOG_IN_REQD
WS_REQUEST_BLOCKED

SOAP Fault Status Codes

The following table defines SOAP fault types and their corresponding codes. For a complete description of faults and how they differ from errors and warnings, refer to Web Services Warnings, Errors, and Faults.

Fault Name	Description
InsufficientPermissionFault	This fault is thrown when the client does not have the appropriate permissions to perform an action based on the role under which they are currently logged in. If the client (user) has more than one role, they may need to login again supplying a different role with more permissions.
InvalidAccountFault	This fault is thrown when the client attempts to login with an invalid account id.
InvalidPartnerCredentials	Partner ID or password submitted with this request is invalid.
InvalidRequestIP	Originating IP is not one of the registered IPs from which this Partner request may come.
InvalidSessionFault	This fault can be thrown when the client's session has timed out or was terminated as the result of a second web services client establishing another session. The fault message occurs on the first request attempted after the session is terminated/timed out. For more details on how NetSuite handles sessions, see Session Management for Web Services.
InvalidVersionFault	This fault is thrown in the event that the request message contains an unsupported version of the schema.

Fault Name	Description
ExceededConcurrentRequest LimitFault	This fault is thrown in the event that the maximum number of concurrent requests has been reached.
ExceededRecordCountFault	This fault is thrown in the event the maximum number of records allowed for an operation has been exceeded. For more information, see Web Services Governance Overview.
ExceededRequestLimitFault	This fault is thrown if the allowed number of concurrent requests is exceeded. For more information, see Web Services Governance Overview.
ExceededRequestSizeFault	
ExceededUsageLimitFault	
UnexpectedErrorFault	This fault is thrown in the event of an occurrence of an unexpected exception.
InvalidCredentialsFault	This fault is thrown in the event of an invalid username (e-mail), password, role, or application ID.
AsyncFault	

Error Status Codes

The following table lists error status code types that can be returned in a message Response. These are values that are used in the code field of the statusDetail type where the type attribute has a value of error.

These codes are listed in the StatusDetailCodeType enumeration, which is defined in the platform faults XSD. You can also view error codes on the Status Detail Code page of the SuiteTalk Schema Browser.

Error Code Returned	Long Description or Message
ABORT_SEARCH_EXCEEDED_ MAX_TIME	This search has timed out. You can choose to schedule it to run in the background and have the results emailed to you when complete. On the saved search form, click the "Email" tab, check "Send According to Schedule", choose an email address on the "Specific Recipients" subtab and a recurrence pattern on the "Schedule" subtab.
ABORT_UPLOAD_VIRUS_ DETECTED	The file {1} contains a virus {2}. Upload abort.
ACCESS_DENIED	Access to this configuration is denied. Please contact NetSuite to gain access to this configuration.
ACCTNG_PRD_REQD	Missing next accounting period
ACCT_DISABLED	account disabled
ACCT_DISABLED	This account has been disabled.
ACCT_DISABLED	Please contact <a href="mailto:{1}"> Accounts Receivable</a> at 650.627.1316 to re-enable this company.
ACCT_DISABLED	Your account has been inactivated by an administrator.
ACCT_MERGE_DUP	The account merge would result in one or more items using duplicate accounts.

Error Code Returned	Long Description or Message
ACCT_NAME_REQD	Accounts require a name.
ACCT_NEEDS_CAMPAIGN_ PROVISION	Please contact your account representative to provision campaign emailing for your account.
ACCT_NOT_CREATED	Account creation was unsuccessful. Technical Support has been alerted to this problem.
ACCT_NUM_REQD	Missing Account Number. Account number is a required field and it cannot be null or empty.
ACCT_NUMS_REQD_OR_ DONT_MATCH	Missing ACCT # or ACCT numbers don't match
ACCT_PERIOD_SETUP_REQD	The accounting period range {1} has not been defined. Please visit '<A href='{2}'>Setup > Accounting > Manage Accounting Periods</A>' to define this period or set up your year.
ACCT_PRDS_BEING_ADDED	Periods are currently being added to this account. Please try again later.
ACCT_REQD	Attempting to adjust provisioning for a customer without an existing account
ACCT_TEMP_DISABLED	You have entered an invalid password on {1} consecutive attempts. Access to your account has been suspended for {2} minutes. If you have forgotten your password, please contact Customer Support.
ACCT_TEMP_UNAVAILABLE	Can't update information - this company's database is currently offline for maintenance. Please try again later.
ACCT_TEMP_UNAVAILABLE	(Temporarily unavailable)
ACCT_TEMP_UNAVAILABLE	The account you are trying to access is currently unavailable while we undergo our regularly scheduled maintenance.
ACCT_TEMP_UNAVAILABLE	We are currently performing maintenance on our system. Please try again soon.
ACCT_TEMP_UNAVAILABLE	The account you are trying to access is currently unavailable while we undergo our regularly scheduled maintenance.
ACCT_TEMP_UNAVAILABLE	Your account is disabled for {1} more minutes due to {2} consecutive failed login attempts.
ACCT_TEMP_UNAVAILABLE	Your account is not yet ready for you to log in. Please wait and try again.
ACCT_TEMP_UNAVAILABLE	Your company database is offline.
ACCT_TEMP_UNAVAILABLE	Your data is still being loaded. Please try again later. Contact <a href='/app/crm/support/nlcorpsupport.nl?type=bug&spf=31'>Professional Services</a> if you have questions.
ACH_NOT_AVAILBL	ACH Processing is not available in this environment.
ACH_SETUP_REQD	Account {1} is not setup for ACH transactions.
ACTIVE_AP_ACCT_REQD	This transaction requires an active Accounts Payable account. Please enable an existing Accounts Payable account, create a new Accounts Payable account, or contact your System Administrator.
ACTIVE_ROLE_REQD	You can only set an active login role as the Web Services default role.
ACTIVE_TRANS_EXIST	There are active direct deposit transactions for this paycheck

Error Code Returned	Long Description or Message
ADDITIONAL_AUTHENTICATION_REQUIRED_2FA	Additional Authentication required - 2FA.
ADDITIONAL_AUTHENTICATION_REQUIRED_SQ	Additional Authentication required - Security Questions.
ADDRESS_LINE_1_REQD	Address Line 1 is a required field and it cannot be null or empty.
ADMIN_ACCESS_REQ	At least one active administrator for each account must have access.
ADMIN_ACCESS_REQ	At least one active administrator for this account must have access.
ADMIN_ACCESS_REQD	Only administrators may enter a memorized transaction in a closed period.
ADMIN_ONLY_ACCESS	{1} only the administrator may access this page.
ADMIN_ONLY_ACCESS	{1} only the administrator may currently access this page.
ADMIN_USER_REQD	User is not an Admin of the demo account
ADMISSIBILITY_PACKG_TYP_REQD	An Admissibility Package Type is required for this international shipment.
ALL_DATA_DELETE_REQD	You must first delete all the data in your account before performing this action. Click <a href='/pages/setup/clearaccount.jsp?import=T'>here</a> to delete your data.
ALL_MTRX_SUBITMES_OPTNS_REQD	The following matrix subitems exist but aren't included in the options you just specified. On the Matrix tab, please make sure the options you select include all existing subitems:<p> {1}
ALREADY_IN_INVT	The following {1} numbers are already in inventory: {2}
ALREADY_IN_INVT	The following {1} number is already in inventory: {2}
AMORTZN_INVALID_DATE_RANGE	Amortization end date can not be before amortization start date.
AMORTZN_TMPLT_DATA_MISSING	One or more line items on this transaction have Variable Amortization Templates, but do not have the required {1} also populated. Please either change the Template for these items or indicate which {1} will be used to schedule the amortization.
AMT_DISALLWD	Description items may not have an amount.
AMT_EXCEEDS_APPROVAL_LIMIT	No one in your chain of command has a sufficient spending limit to approve this transaction.
ANSWER_REQD	Please provide an answer.
APPROVAL_PERMS_REQD	{1} The restrictions on your role do not allow you to approve or reject this record.
AREA_CODE_REQD	Please include an area code with the phone number.
ASSIGNEE_REQD	{1} must be assigned to {2}
AT_LEAST_ONE_FILE_REQD	Download folder must have at least one file.
AT_LEAST_ONE_PACKAGE_REQD	1 or more packages are required.

Error Code Returned	Long Description or Message
AT_LEAST_ONE_RETURN_FLD_REQD	You must specify at least one field to be returned.
AT_LEAST_ONE_SUB_REQD	You must choose at least one subsidiary.
ATTACH_SIZE_EXCEEDED	The data you are uploading exceeds the maximum allowable size of {1}. Please change your selection and try again
ATTACH_SIZE_EXCEEDED	This message exceeds the limit of 24 MB. Please reduce the size of the message and its attachments and try again.<br/>Note: Files can be larger when attached due to encoding.
ATTACHMNT_CONTAINS_VIRUS	The attachment with file name {1} contains a virus {2}. It is removed from the message.
ATTACHMNT_CONTAINS_VIRUS	The attachment file {0} contains virus {1}. Save message abort.
AUDIT_W2_1099	Use this report to audit the information that will be used to generate W2s and 1099s.
AUTO_NUM_UPDATE_DISALLWD	We currently do not support an automatic numbering update of more than {1} {2} records. Please contact <A href='/app/crm/support/nlcorpsupport.nl?type=support'>NetSuite support</A> to request a full numbering update of your {2}s.</a>
AVS_ERROR	This order failed the AVS check and will not be saved. An AVS mismatch has been detected and the transaction has been rejected. However, a valid authorization has been created against the cardholder's account. If you want to accept this order, click Go Back and check the Ignore AVS checkbox and Save the order. Note that this will result in a second authorization.
BALANCE_EXCEEDS_CREDIT_LIMIT	Customer balance exceeds credit limit
BANK_ACCT_REQD	You must have a bank account to perform this operation. Click <a href='/app/accounting/account/account.nl'>here</a> to add one.
BASE_CRNCY_REQD	You may not delete you base currency.
BILL_PAY_STATUS_UNAVAILABLE	View Online Bill Pay Status information is currently not available. Please try again in a few minutes.
BILL_PAY_STATUS_UNAVAILABLE	View Online Bill Pay Status is not available until your bill pay registration is complete.
BILL_PMTS_MADE_FROM_ACCT_ONLY	Your payment has been recorded, but online bill payments can only be made from the account <b>{1}</b>, so no online bill pay payment will be made. You should return to the payment screen if you wish to print the check.
BILLABLES_DISALLWD	{1} does not allow billables.
BILLING_ISSUES	Your account has been locked due to billing issues. You must call your NetSuite Sales Representative for further assistance.
BILLING_ISSUES	Your account has not been fully paid for. Please log in to your account and follow the billing process or contact your Account Manager.
BILLING_SCHDUL_INVALID_RECURR	Billing schedules may not have a recurrence count greater than 500

Error Code Returned	Long Description or Message
BILLPAY_APPROVAL_UNAVAILBL	Approve Online Bill Payments is currently not available. Please try again in a few minutes.
BILLPAY_REGSTRTN_REQD	Online bill pay approve payments is not available until your billpay registration is complete.
BILLPAY_SRVC_UNAVAILBL	Online Bill Pay service is temporarily suspended. If you prefer to wait until Online Bill Pay service is restored, just leave payments to be approved in this list. We will notify you by email as soon as the service is available again. If you need to make an urgent payment, we suggest that you print a check. To do this: 1. Clear the Online Bill Pay check box for that payment. 2. Click the underlined date of the payment, and you will return to the Bill Payment page. 3. On the Bill Payment page, click To Be Printed instead of Bill Pay. 4. Click Submit. 5. Go to Transactions > Print Checks and Forms > Checks, and then mark the check to be printed. 6. Click Submit.
BILLPAY_SRVC_UNAVAILBL	The Online Bill Pay service is currently not available. Please try again in a few minutes.
BIN_DSNT_CONTAIN_ENOUGH_ITEM	The following bins do not contain enough of the requested item ({1}): {2}
BIN_DSNT_CONTAIN_ENOUGH_ITEM	The following bin does not contain enough of the requested item ({1}): {2}
BIN_ITEM_UNAVAILBL	The following bins are not available for the specified item: {1}
BIN_ITEM_UNAVAILBL	The following bins are not available for the specified item ({1}): {2}
BIN_ITEM_UNAVAILBL	The following bin in not available for the specified item ({1}): {2}
BIN_ITEM_UNAVAILBL	The following bin is not available for the specified item: {1}
BIN_SETUP_REQD	The following bins are not associated with the item '{1}': {2}. You can associate bins with an item on the inventory tab of the item record.
BIN_UNDEFND	The following bins specified for the item {1} are not defined in the transaction location ({2}): {3}
BUNDLE_IS_DEPRECATED	This bundle is no longer available. It has been deprecated by bundle {1} on account {2}.
BUNDLE_IS_DEPRECATED	You cannot update this bundle as it has been deprecated and you have not been granted access to the replacement bundle. Please contact the solution provider.
CALENDAR_PREFS_REQD	Set up {1} Calendar Preferences first.
CALENDAR_PREFS_REQD	Set up Calendar Preferences first
CAMPAGIN_ALREADY_EXECUTED	You cannot delete email campaigns that have already been executed
CAMPAIGN_IN_USE	You cannot delete a campaign event that already has activity.
CAMPAIGN_SET_UP_REQD	The following steps need to be performed before a campaign can be created:<p>{1}
CANNOT_RESET_PASSWORD	Sorry, we are currently unable to reset your password. Please contact support or try again later.
CANT_APPLY_PMT	The top level entity cannot accept payment because it has a status of {1}. It must have a status of {2} or {3} to accept payment.

Error Code Returned	Long Description or Message
CANT_APPLY_PMT	This entity cannot accept payment because it has a status of {1}. It must have a status of {2} or {3} to accept payment.
CANT_AUTO_CREATE_ ADJSTMNT	The 'Intercompany Expenses' accounting preference does not currently support the automated creation of adjustments. If you would like to create adjustments automatically, please change that preference to 'Allow and Auto Adjust.'
CANT_CALC_FEDEX_RATES	FedEx rates cannot be calculated:
CANT_CANCEL_APPRVD_ RETRN_AUTH	You cannot cancel this return authorization because it has already been approved.
CANT_CANCEL_BILL_PMT	The Online Bill Payment cannot be stopped because the payment may already have been made.
CANT_CHANGE_COMMSSN	The Online Bill Payment cannot be stopped because the payment may already have been made.
CANT_CHANGE_CONTACT_ RESTRICTN	You cannot change the restriction on this contact.
CANT_CHANGE_ CRMRECORDTYPELINKS	Cannot alter standard CrmRecordTypeLinks
CANT_CHANGE_EVENT_ PRIMARY_TYP	You cannot change the primary type for this event
CANT_CHANGE_IP_ADDRESS	The domain {1} is currently associated with IP address {2}. You cannot change the IP address of a live domain.
CANT_CHANGE_LEAD_ SOURCE_CAT	You cannot change the category for a leadsource that is defined as the default leadsource for another category
CANT_CHANGE_PSWD	You changed your password less than 24 hours ago. NetSuite only allows one password change per 24-hour period.
CANT_CHANGE_REV_REC_ TMPLT	The rev rec template on a billable expense can not be changed or removed after it is saved.
CANT_CHANGE_REV_REC_ TMPLT	The rev rec template on billable time and items can not be changed or removed after it is saved.
CANT_CHANGE_SUB	You cannot change the subsidiary on this record because doing so will change the subsidiary selected on the associated employee record.
CANT_CHANGE_TASK_LINK	Cannot alter standard task links
CANT_CHANGE_UNITS_TYP	You may not change the units type of an item after it has been set.
CANT_CHANGE_VSOE_ ALLOCTN	You are attempting to change the VSOE Allocation for a transaction in a closed period. You must either change the posting period for the related transaction or open the period.
CANT_CHG_POSTED_BILL_ VRNC	The receipts for this bill can not be changed after the bill variance has been posted.
CANT_CHG_POSTED_BILL_ VRNC	One or more lines have had their bill variance posted and can not be changed.
CANT_COMPLETE_FULFILL	The fulfillment cannot be completed.
CANT_CONNECT_TO_STORE	Error - Unable to connect to store {1}
CANT_CONVERT_CLASS_ DEPT	You cannot convert classes to departments because you already have department data in the system. You must remove all departments first.

Error Code Returned	Long Description or Message
CANT_CONVERT_CLASS_LOC	You cannot convert classes to locations because you already have location data in the system. You must remove all locations first.
CANT_CONVERT_INVT_ITEM	This item is a member of a kit or a non-numbered assembly. You may not convert it to a numbered inventory item.
CANT_CREAT_SHIP_LABEL	A shipping label could not be generated because the In Bond Code field is not set. Please enter a value in the In Bond Code field on the Item Fulfillment page.
CANT_CREATE_FILES	Could not create files for uploading your data
CANT_CREATE_NON_ UNIQUE_RCRD	A record with the same unique signatures already exists. You must enter unique signatures for each record you create.
CANT_CREATE_PO	Purchase Orders cannot be created for assembly items.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because a currency must be defined for the "Ship From" country "{1}" when using the Insured Value option. Go to Lists -> Accounting -> Currencies to create a currency for {1}.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because a currency must be defined for the "Ship To" country "{1}" when using the COD option. Go to Lists -> Accounting -> Currencies to create a currency for {1}.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the Addressee field of the "Ship To" address is not set. Please enter a "Ship To" Addressee on the Item Fulfillment page.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the Address 1 field of the "Ship From" address is not set. Please go to $(regex) to enter the "Ship From" Address 1.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the Address 1 field of the "Ship To" address is not set. Please enter a "Ship To" Address 1 on the Item Fulfillment page.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the Attention field of the "Ship From" address is not set. Please go to $(regex) to enter the "Ship From" Attention.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the City field of the "Ship To" address is not set. Please enter a "Ship To" City on the Item Fulfillment page.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the City of the "Ship From" address is not set. Please go to $(regex) to enter the "Ship From" City.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the Company or Location Name of the "Ship From" address is not set. Please go to $(regex) to enter the "Ship From" Company/Location Name.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the Country field of the "Ship To" address is not set. Please enter a "Ship To" Country on the Item Fulfillment page.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the Country of the "Ship From" address is not set. Please go to $(regex) to set the "Ship From" Country.

Error Code Returned	Long Description or Message
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the Package Weight was not entered. Please enter a value in the Package Weight field on the Item Fulfillment page.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the Phone Number of the "Ship From" address is not set. Please go to $(regex) to set the "Ship From" Phone Number.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the Phone Number of the "Ship To" address is not set. Please enter a "Ship To" Phone Number on the Item Fulfillment page.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the Pickup Type was not set. Please go to Setup > Set Up Shipping to select a shipping Pickup Type.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the shipping method was not set. Please go to Lists > Shipping Items to select a Shipping Label Integration shipping method for this shipping item.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the State field of the "Ship To" address is not set. Please enter a "Ship To" State on the Item Fulfillment page.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the State of the "Ship From" address is not set. Please go to $(regex) to enter the "Ship From" State.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the Zip Code of the "Ship From" address is not set. Please go to $(regex) to enter the "Ship From" Zip Code.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because the Zip field of the "Ship To" address is not set. Please enter a "Ship To" Zip code on the Item Fulfillment page.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because your {1} Account Number is not set. Go to Setup > Set Up Shipping > {2} Registration to enter your {3} Account Number.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because your {1} Registration Address Line 1 is not set. Go to Setup > {2} Registration to complete the Address Line 1 field.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because your {1} Registration City is not set. Go to Setup > {2} Registration to enter your City.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because your {1} Registration Company field is not set. Go to Setup > {2} Registration to enter a name in the Company field.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because your {1} Registration Country is not set. Go to Setup > {2} Registration to select your Country.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because your {1} Registration Ship to Attention field is not set. Go to Setup > {2} Registration to enter a name in the Ship to Attention field.
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because your {1} Registration State is not set. Go to Setup > {2} Registration to select or enter your State.

Error Code Returned	Long Description or Message
CANT_CREATE_SHIP_LABEL	A shipping label could not be generated because your {1} Registration Zip Code is not set. Go to Setup > {2} Registration to enter your Zip Code.
CANT_CREATE_WORK_ORD	Work orders can only be created for assembly items.
CANT_DEL_DEFAULT_CALENDAR	
CANT_DEL_DEFAULT_SHIP_METHOD	This Shipping Item cannot be deleted because it is the Default Shipping Method. Please go to Setup > Accounting > Set Up Shipping and choose a new Default Shipping Method before deleting this Shipping Item.
CANT_DEL_REALIZED_GAINLOSS	A Realized Gain/Loss Transaction cannot be deleted.
CANT_DEL_TRANS_RVRSL	The reversal of the month-end Unrealized Gain/Loss transaction cannot be deleted.
CANT_DELETE_ACCT	This account cannot be deleted because it has associated transactions.
CANT_DELETE_ACCT	This account cannot be deleted because it is a special type of account needed by {1}
CANT_DELETE_ACCT	This account cannot be deleted because it is a special type of account needed by NetSuite
CANT_DELETE_ACCT	This account cannot be deleted because it is a special type of account needed by the system.
CANT_DELETE_ACCT	This account cannot be deleted because it is used by one or more transactions or it has child accounts or it is used by one or more items.
CANT_DELETE_ACCT_PRD	You may not delete an accounting period with transactions posted to it. You must first edit the transactions, change the posting period and then delete the period.
CANT_DELETE_ALLOCTN	This allocation detail can not be deleted because it has a journal entry.
CANT_DELETE_BIN	You may not delete this bin record because it is already in use. You must either remove all references to it in item records and transactions or make it inactive.
CANT_DELETE_CATEGORY	This category cannot be deleted because it has child items
CANT_DELETE_CATEGORY	This category cannot be deleted because it has subcategories
CANT_DELETE_CC_PROCESSOR	This credit card processor is used in transaction and cannot be deleted.
CANT_DELETE_CELL	This cell cannot be deleted because it has child items
CANT_DELETE_CHILD_RCRD_FOUND	This {1} record cannot be deleted because it is referenced by other records.
CANT_DELETE_CHILD_RCRDS_EXIST	This record can not be deleted because it has child records.
CANT_DELETE_CLASS	This class cannot be deleted because it has child items
CANT_DELETE_COLOR_THEME	This color theme cannot be deleted because it is being used
CANT_DELETE_COMMSSN_SCHDUL	This schedule has already been used to generate commission calculations and can't be deleted. If no authorizations have been made, schedule can be deleted after being removed from all active plans.

Error Code Returned	Long Description or Message
CANT_DELETE_COMPANY	This company cannot be deleted because it has child entities
CANT_DELETE_COMPANY_TYP	This company type cannot be deleted because the company has associated transactions.
CANT_DELETE_CONTACT_HAS_CHILD	The contact record cannot be deleted because it has child records.
CANT_DELETE_CONTACT_HAS_CHILD	This contact cannot be deleted because it has child entities
CANT_DELETE_CSTM_FIELD	This custom field cannot be deleted because it is referred to by other custom fields
CANT_DELETE_CSTM_FORM	This custom form cannot be deleted because it is referred to by other custom forms
CANT_DELETE_CSTM_ITEM_FIELD	This custom item field has dependent matrix items. It can not be deleted.
CANT_DELETE_CSTM_LAYOUT	This custom layout cannot be deleted because it is used by custom forms
CANT_DELETE_CSTM_LIST	This custom list cannot be deleted because it is referred to by custom fields
CANT_DELETE_CSTM_RCRD	This custom record cannot be deleted because it is referred to by custom fields
CANT_DELETE_CSTM_RCRD_ENTRY	This custom record entry cannot be deleted because it is referred to by other records
CANT_DELETE_CUST	You can't delete this customer because it's set up as default Anonymous Customer
CANT_DELETE_CUSTOMER	This customer or job cannot be deleted because it has child entities.
CANT_DELETE_DEFAULT_FLDR	You cannot delete the default folders.
CANT_DELETE_DEFAULT_PRIORITY	You cannot delete the default case priority. Please select a new default first.
CANT_DELETE_DEFAULT_SALES_REP	Default Sales Rep Role cannot be deleted.
CANT_DELETE_DEFAULT_STATUS	You cannot delete a default case status. Please select a new default first.
CANT_DELETE_DEFAULT_STATUS	You can't delete or inactivate that status because it is a set up as a default status. Please navigate to <a href='/app/setup/sfasetup.nl' target='_blank'>Sales Preferences</a> and change that status
CANT_DELETE_DEFAULT_VALUE	You may not delete or inactivate that value because it is a default. Please select a new default first.
CANT_DELETE_DEFAULT_WEBSITE	The default Web site cannot be deleted.
CANT_DELETE_EMPL	This employee cannot be deleted because it has child entities
CANT_DELETE_ENTITY	This entity cannot be deleted because it has child items
CANT_DELETE_FIN_STATMNT_LAYOUT	This financial statement layout cannot be deleted because it is referred to by other layouts.

Error Code Returned	Long Description or Message
CANT_DELETE_FLDR	These predefined folders cannot be deleted
CANT_DELETE_HAS_CHILD_ ITEM	This {1} cannot be deleted because it has child items
CANT_DELETE_INFO_ITEM	This information item cannot be deleted because it has child items
CANT_DELETE_ITEM	This item cannot be deleted because it has child items
CANT_DELETE_ITEM_ LAYOUT	This item/category layout cannot be deleted because it is used by store tabs
CANT_DELETE_ITEM_TMPLT	This item/category template cannot be deleted because it is referred to by a theme or an item
CANT_DELETE_JOB_ RESOURCE_ROLE	Default Job Resource Role cannot be deleted.
CANT_DELETE_LEGACY_ CATEGORY	Legacy category cannot be removed
CANT_DELETE_LINE	This line cannot be deleted, because it is referred to by other records. Before removing this line, remove any discount or markup lines applied to it.
CANT_DELETE_MEDIA_ITEM	This media item cannot be deleted because it is being referenced by another item.
CANT_DELETE_MEMRZD_ TRANS	This memorized transaction cannot be deleted because it referenced in transactions
CANT_DELETE_OR_CHANGE_ ACCT	Special accounts cannot be deleted and their type cannot be changed
CANT_DELETE_PLAN_ ASSGNMNT	Trying to delete plan assignment referenced by precalcs.
CANT_DELETE_PRESNTN_ CAT	This presentation category cannot be deleted because it has subcategories
CANT_DELETE_RCRD	This {1} record cannot be deleted because it referenced by other records
CANT_DELETE_RCRD	This record cannot be deleted because it has {1}child records{2}
CANT_DELETE_RCRD	This record cannot be deleted because it is referenced by other records or it is used by one or more transactions.
CANT_DELETE_RCRD	This record cannot be deleted, because it is referred to by other records.
CANT_DELETE_RCRDS	Selected records could not be deleted because one or more of them are of a special type of account needed by {1}
CANT_DELETE_RCRDS	Selected records could not be deleted because one or more of them are referenced by other records.
CANT_DELETE_SITE_TAG	This site tag cannot be deleted because it is being used.
CANT_DELETE_SITE_THEME	This site theme cannot be deleted because it is being used
CANT_DELETE_SOLUTN	This solution cannot be deleted because it has been applied to support cases.
CANT_DELETE_STATUS_TYPE	You cannot delete the only status of type {1}
CANT_DELETE_SUBTAB	This subtab cannot be deleted because it is referred to by custom fields

Error Code Returned	Long Description or Message
CANT_DELETE_SYSTEM_ NOTE	You cannot alter or delete a system logged note.
CANT_DELETE_TAX_VENDOR	This is a special tax vendor and cannot be deleted.
CANT_DELETE_TMPLT_RCRD	This template record cannot be deleted.
CANT_DELETE_TRANS	This transaction cannot be deleted because it is a Google Checkout order awaiting updated payment information.
CANT_DELETE_TRANS	This transaction cannot be deleted because it is a posting Google Checkout order.
CANT_DELETE_TRANS	This transaction cannot be deleted because it is linked to one or more commission transactions. The commission authorizations due to this transaction need to be removed to be able to delete this transaction.
CANT_DELETE_TRANS	This transaction cannot be deleted because it is referenced by an intercompany adjustment. The adjustment must be deleted first.
CANT_DELETE_TRANS	This transaction cannot be deleted because it is referred to by other transactions. It may be a bill or an invoice that has been paid or an expense that has been reimbursed.
CANT_DELETE_TRAN_LINE	Failed to delete line {1}. This line is linked to another transaction.
CANT_DELETE_TRAN_LINES	Lines with partially recognized rev rec or amortization schedules can not be deleted.
CANT_DELETE_UPDATE_ ACCT	This account cannot be deleted or changed because it is a special type of account needed by {1}
CANT_DELETE_URL	This third party conversion tracking URL cannot be deleted because it is reference by other records.
CANT_DELETE_VENDOR	This {1:Vendor} is related to a {2:Partner} which is eligible for commission and cannot be deleted.
CANT_DELETE_VENDOR	This vendor cannot be deleted because there are dependent items, such as a pending payment. If you wish to remove the payee, you must first delete all such dependent items.
CANT_DIVIDE_BY_ZERO	There is a divide by zero error in this search. It may be an error with a formula you have used. Please retry without the formula(s). If an error still occurs, please file with Customer Support. If it does not, please correct your formula. Typically that consists of taking the denominator and wrapping it in NULLIF(<denominator>,0).
CANT_DOWNLOAD_ EXPIRED_FILE	This file has expired and can no longer be downloaded
CANT_EDIT_CHARGED_ ORDER	You cannot charge an order that is already completely charged.
CANT_EDIT_CUST_LIST	Can not modify this Custom List because its entries are in use.
CANT_EDIT_CUST_PMT	This customer payment cannot be edited while it has an Automated Clearing House transmission in process.</TD></TR><TR><TD class=text> </TD></TR><TR><TD class=text> To view the status of customer payments with ACH transmissions, go to Transactions > View Electronic Funds Transfer Status.
CANT_EDIT_DPLYMNT_IN_ PROGRESS	You cannot change or delete a deployment that is in progress or in the queue.

Error Code Returned	Long Description or Message
CANT_EDIT_DPLYMNT_IN_ PROGRESS	You cannot edit a script deployment when it is being executed.
CANT_EDIT_FOLDER	Predefined folders cannot be updated.
CANT_EDIT_OLD_CASE	This case cannot be edited because it was closed {1} or more days ago.
CANT_EDIT_STANDARD_OBJ	Cannot alter standard dashboards
CANT_EDIT_STANDARD_OBJ	Cannot alter standard dashboard role maps
CANT_EDIT_STANDARD_OBJ	Cannot alter standard dashboard section maps
CANT_EDIT_STANDARD_OBJ	Cannot alter standard fields
CANT_EDIT_STANDARD_OBJ	Cannot alter standard forms or layouts
CANT_EDIT_STANDARD_OBJ	Cannot alter standard portlets
CANT_EDIT_STANDARD_OBJ	Cannot alter standard report snapshot layouts
CANT_EDIT_STANDARD_OBJ	Cannot alter standard roles
CANT_EDIT_STANDARD_OBJ	Cannot alter standard searches
CANT_EDIT_STANDARD_OBJ	Cannot alter standard sections
CANT_EDIT_STANDARD_OBJ	Cannot alter standard tabs
CANT_EDIT_STANDARD_OBJ	Cannot alter standard tasks
CANT_EDIT_STANDARD_OBJ	Cannot alter standard task categories
CANT_EDIT_STANDARD_OBJ	Cannot alter standard templates
CANT_EDIT_STANDARD_OBJ	Cannot Alter Standard Types
CANT_EDIT_STANDARD_OBJ	Cannot alter standard words
CANT_EDIT_TAGATA	The Receivable Tegata is linked to Invoices and is no longer editable
CANT_EDIT_TRAN	You cannot edit intercompany adjustments.
CANT_EDIT_TRAN	You cannot edit the account of a transaction line that is linked to others
CANT_EDIT_TRAN	You cannot edit this expense report
CANT_EDIT_TRAN	You cannot edit this transaction because it was automatically created by the NetSuite Payroll Service
CANT_ESTABLISH_LINK	Unable to establish link with {1}
CANT_FIND_BUG	Cannot locate the bug that was just entered (1)!
CANT_FIND_MAIL_MERGE_ID	Mail Merge Id not found
CANT_FIND_RCRD	Could not find record with {1} = {2}
CANT_FIND_SAVED_IMPORT	No saved import with internalId {1}
CANT_FIND_SOURCE_ AMORTZN_ACCT	The source account for the amortization schedule could not be determined.
CANT_FIND_UPS_REG_FOR_ LOC	No UPS registration was found for the location selected. Please select a different shipping item, or go to Setup > Set Up Shipping to register a UPS account for this location.
CANT_FULFILL_ITEM	An item receipt has been posted

Error Code Returned	Long Description or Message
CANT_INACTIVATE_COMMSSN_PLAN	You cannot inactivate a plan that has commission payments that are pending authorization. Please clear the commission payments at Transactions > Authorize Commissions before inactivating this plan.
CANT_INACTIVE_DEFAULT_SYNC_CAT	You cannot inactivate the default synchronization category.
CANT_INACTIVE_DEFAULT_TMPLT	You cannot inactivate this template record because it is set up as a default template
CANT_LOAD_SAVED_SEARCH_PARAM	Error loading saved search params
CANT_LOGIN_WITH_OAUTH	A login operation or Request Level Credentials must not be used in conjunction with OAuth authorization{:do not translate 'login' or capitalized words}
	Can not lookup field {1} by {2}
CANT_MAKE_CONTACT_PRIVATE	Employee contacts cannot be made private
CANT_MAKE_CONTACT_PRIVATE	Individual relationship contacts cannot be made private
CANT_MARK_SHIPPED	Item {1:item name} cannot be marked shipped because the remaining quantity on the linked purchase order does not match the remaining quantity on the sales order.
CANT_MERGE_EMPLS	employees can not be merged
CANT_MODIFY_APPRVD_TIME	Time records can not be modified after they have been approved.
CANT_MODIFY_FULFILL_STATUS	You may not change the fulfillable status of an item which has transactions associated with it.
CANT_MODIFY_ISSUE_STATUS	The issue status '{1:issue status name}' cannot be changed from {2:base status} to {3:base status} because it is in use.
CANT_MODIFY_LOCKED_FLD	You may not update or delete a locked custom field.
CANT_MODIFY_PARENT	Payments have been accepted from the top level parent. The top level parent can not be changed.
CANT_MODIFY_REV_REC	The value of Rev Rec on Rev Commit may not be modified for this transaction.
CANT_MODIFY_SUB	You cannot change the subsidiary of this entity because one or more transactions exist for this entity.
CANT_MODIFY_TAGATA	The Payable Tegata is no longer in Issued state and cannot be modified.'
CANT_MODIFY_TAGATA	The Receivable Tegata is no longer in Holding state and cannot be modified.
CANT_MODIFY_TEGATA	The Payable Tegata is linked to bills and cannot be modified.'
CANT_MODIFY_VOID_TRANS	The G/L impact of a voided transaction cannot be changed.
CANT_MODIFY_VOID_TRANS	You may not change the GL impact on a voided {1: transaction type}.
CANT_MOVE_REALIZED_GAINLOSS	You cannot move a Realized Gain/Loss transaction to a date before either the source or the payment transaction.
CANT_PAY_TAGATA	Endorsed Tegata can only be paid on or after its maturity date.

Error Code Returned	Long Description or Message
CANT_PAY_TAGATA	Payable Tegata can only be paid on or after its maturity date.
CANT_PROCESS_IMG	Faceless PDF Library unable to process image. Image DPI:{1}.
CANT_RCEIV_BEFORE_FULFILL	The item receipt for a transfer order line can not occur before the item fulfillment.
CANT_RCEIV_ITEM	A mark shipped fulfillment line has been processed against item {1:item name}. This item cannot be received.
CANT_RECEIVE_TAGATA	Receivable Tegata can only be collected on or after its maturity date.
CANT_REJECT_ORDER	You cannot reject this order because it has already been approved.
CANT_REMOV_ALL_FULFILMNT_LINKS	You may not modify this sales order in such a way that it removes all links to any fulfillment. The modifications you made would leave the fulfillment <a href="/app/accounting/transactions/transaction.nl?id={1}">{2}</a> unlinked.
CANT_REMOV_ITEM_SUB	You may not remove a subsidiary from an item that is a member of an assembly, group, or kit item if the parent item is available in that subsidiary.
CANT_REMOVE_ACH_PAY_METHOD	ACH payment methods cannot be removed
CANT_REMOVE_APPROVAL	The accounting approval cannot be removed from this expense report because some of its lines have already been invoiced to the customer.
CANT_REMOVE_DOMAIN	You are trying to remove a domain that is referenced by {1} CRM template(s). Please first clear the domain from CRM templates before trying to remove it.
CANT_REMOVE_NEXUS	A nexus cannot be removed from a subsidiary if the nexus is associated with a transaction.
CANT_REMOVE_SCHDUL	You have attempted to remove an active schedule from a plan. Removing this participant is not permitted after commissions against the plan have been generated.
CANT_REMOVE_SUB	You cannot remove subsidiary: {1} because this record is used on a transaction for subsidiary: {1}.
CANT_REMOVE_SUB	You attempted to remove one or more subsidiaries from this item, but the item appears in at least one transaction in those subsidiaries. To remove a subsidiary from the item, make sure the item does not appear in any transactions for that subsidiary.
CANT_RESUBMIT_FAILED_DPLYMNT	You cannot submit a deployment for execution whose status is set to Scheduled.
CANT_RETURN_FLD	Can not return field {1}. Reason: {2}
CANT_RETURN_USED_GIFT_CERT	Used gift certificates can not be returned.
CANT_REV_REC_BODY_AND_LINE	The Revenue Recognition fields must be specified at EITHER the transaction body or the item line level, and may NOT be specified at both levels.
CANT_REVERSE_AUTH	Card type doesn't allow reversals. No resolution. Authorization cannot be reversed.
CANT_SCHDUL_RECUR_EVENT	Because the number of days in each month differs, recurring monthly events cannot be scheduled after the 28th.

Error Code Returned	Long Description or Message
CANT_SEND_EMAIL	Unable to send notification email
CANT_SEND_EMAIL	Unable to send notification email to support rep
CANT_SET_CLOSE_DATE	Unable to set expected close date of prospect/lead based on current estimates/opportunities.
CANT_SET_INTERNALID	You cannot set internalId with upsert.
CANT_SET_STATUS	Unable to set status of prospect/lead based on current estimates.
CANT_SWITCH_ROLES_FROM_LOGIN	Role switching is not allowed from this login.
CANT_SWITCH_SHIP_METHOD	Switching the shipping method to another carrier is an unsupported operation, because it requires reloading the item fulfillment form for that carrier.
CANT_UPDATE_ACCTNG_PRDS	You cannot update accounting periods using SuiteScript or Web Services. Go to Setup > Accounting > Manage G/L > Manage Accounting Periods.
CANT_UPDATE_AMT	The amount on lines containing partially/fully recognized schedules can not be changed
CANT_UPDATE_DYNAMIC_GROUP	You cannot update dynamic groups. Instead you must modify the saved search associated with the group
CANT_UPDATE_FLDR	These predefined folders cannot be updated
CANT_UPDATE_LINKED_TRANS_LINES	You cannot update linked transaction lines
CANT_UPDATE_PRODUCT_FEED	This item has multiple product feeds. Web Services schema version 2_6 or greater is required to modify product feeds for this item
CANT_UPDATE_RECRD_HAS_CHANGED	Cannot update bug. Record has changed since you last retrieved it.
CANT_UPDATE_RECUR_EVENT	Event <id {1}> contains recurrence patterns that are not supported in your client application. You are not allowed to update recurrence pattern on this event. Contact your software vendor for the latest Web Services upgrade.
CANT_UPDATE_ROOT_CATEGORY	Can not update root level website categories through Web Services.
CANT_UPDATE_STATUS_TYPE	You cannot update the only status of type {1}
CANT_VERIFY_CARD	Card Verify not supported. Retry request.
CANT_VOID_TRANS	You cannot void this transaction because it is linked to by one or more transactions such as payments. You must delete or void those transactions first
CARD_EXPIRED	Expired Card. Re-submit with valid expiration date.
CARD_ID_REQD	Card ID required. Provide a valid card ID.
CASE_ALREADY_ASSIGNED	This case cannot be grabbed because it is already assigned to another rep. To view the case, go back and click on the case number.
CASE_DSNT_EXIST	Case doesn't exist or no customer is associated with case.
CASE_NOT_GROUP_MEMBER	{1} this case record does not belong to your group.

Error Code Returned	Long Description or Message
CASH_SALE_EDIT_DISALLWD	This cash sale cannot be edited while it has an Automated Clearing House transmission in process.</TD></TR><TR><TD class=text> </TD></TR><TR><TD class=text> To view the status of cash sales with ACH transmissions, go to Transactions > View Electronic Funds Transfer Status.
CC_ACCT_REQD	You must have a credit card account to perform this operation.
CC_ACCT_REQD	You must have a credit card account to perform this operation. Click <a href='/app/accounting/account/account.nl'>here</a> to add one.
CC_ALREADY_SAVED	That credit card is already saved. Please use the saved credit card.
CC_EMAIL_ADDRESS_REQD	Please go back and provide an email address to CC store orders to.
CC_NUM_REQD	Please provide a credit card number.
CC_PROCESSOR_ERROR	An error occurred while processing the credit card. Please contact the merchant for assistance.
CC_PROCESSOR_ERROR	An unexpected error occurred while processing the credit card through Merchant e-Solutions (reason code = {1}). Please contact NetSuite support.
CC_PROCESSOR_NOT_FOUND	A suitable credit card processor was not found for this transaction.
CC_SECURITY_CODE_REQD	This transaction requires the Credit Card Security Code. Please enter the required value in the {1} field and re-submit.
CERT_AUTH_EXPD	CA expired on {1}
CERT_EXPD	Certificate expired on {1}
CERT_UNAVAILABLE	Certificate unavailable (most likely has not been presented by client)
CHANGE_PMT_DATE_AND_REAPPROVE	The payment is more than 30 days past due and has NOT been sent. Edit the payment to change the date and reapprove.
CHAR_ERROR	Character error on Line# {1} Column# {2} (Byte # {3}). {4}
CHECKOUT_EMAIL_REQD	Please go back and provide an email address to email checkout errors to.
CITY_REQD	City is a required field and it cannot be null or empty.
CLASS_ALREADY_EXISTS	A class already exists with that name. Go <a href="javascript:history.go(-1);";>back</a>, change the name and resubmit.
CLASS_OR_DEPT_OR_CUST_REQD	only one of class, cust, and dept can be non-null
CLEAR_AUTOCALC	For items that use the time phased replenishment method, you must clear the Auto-Calculate checkbox next to the Reorder Point and Preferred Stock Level fields. The mass update cannot be performed unless these settings are changed.
CLOSE_PREVIOUSE_PERIOD	Please close previous period before working on this one.
CLOSED_TRAN_PRD	The G/L impact of a transaction in a closed period cannot be changed.
CLOSED_TRAN_PRD	You cannot move a transaction to or from a closed period.
COGS_ERROR	COGS lines not cleaned up
COGS_ERROR	Cost of Goods Sold lines not in balance

Error Code Returned	Long Description or Message
COGS_ERROR	LIFO/FIFO COGS count does not equal the number of items requested COGS ERROR 9765 itemsLinked={1}, itemsTotal={2} kdoc={3}, nid={4}
COMMSSN_ALREADY_ CALCLTD	You have attempted to remove an active sales participant from a plan. Removing this participant is not permitted after commissions against the plan have been generated.
COMMSSN_FEATURE_ DISABLED	You have not enabled the Commissions feature.
COMMSSN_FEATURE_ DISABLED	You have not enabled the Partner Commissions/Royalties feature.
COMMSSN_PAYROLL_ITEM_ REQD	A commission payroll item must be added for each employee to be processed through payroll
COMP_DELETED_OR_ MERGED	The company you try to attach the context to has been deleted or merged.
COMPANION_PROP_REQD	Error - Items do not have companion property (column) {1}
COMPANY_FLD_REQD	The Company field is required for COD shipments. Go to Setup > Accounting > Shipping, set the Company field on your FedEx Registration record and re-submit.
CONCUR_BILLPAY_JOB_ DISALLWD	Your account currently has a bill pay approval job in progress. Only one bill pay approval job per account is allowed at a time. Please wait until this process completes before submitting another group of payments for approval. <BR><BR>Visit the <a href='/app/external/xml/upload/uploadl og.nl?displayType=BILLPAY'>status page </a> to track the progress of the current job.
CONCUR_BULK_JOB_ DISALLWD	This Account is already running a bulk processing job. Please visit the <a href='/app/external/xml/upload/uploadlog.nl?displayType=BULKFULFILL' >status page </a> to track the progress of the current job.
CONCUR_MASS_UPDATE_ DISALLWD	A mass update is currently running in this account. Please try again in a few minutes.
CONCUR_SEARCH_ DISALLWD	Search aborted by concurrent {1} search. Only one search may run at a time.
CONSLD_PRNT_AND_CHILD_ DISALLWD	A company can be a consolidated child or a consolidated parent but not both
CONTACT_ALREADY_EXISTS	A contact record with this name already exists. Every contact record must have a unique name.
CONTACT_ALREADY_EXISTS	A contact with the name [{1}] already exists
CONTACT_NOT_GROUP_ MEMBR	{1} this contact does not belong to your group.
COOKIES_DISABLED	You have disabled cookies from being stored on your computer or turned off per-session cookies. Please enable this feature and try again
COUNTRY_STATE_MISMATCH	The country and state/province are mismatched, the country is {1} and the state/province is {2}. Please enter a state/province short name that matches the country.
CREATEDFROM_REQD	Please enter a value for createdFrom.
CREDITS_DISALLWD	Credits Not Allowed. Contact Merchant e-Solutions to have credits enabled.

Error Code Returned	Long Description or Message
CRNCY_MISMATCH_BASE_CRNCY	The currency you are registered to use is different from the base currency of this company.
CRNCY_NOT_UPDATED	The following currencies were not updated: {1}
CRNCY_RCRD_DELETED	This currency record has been deleted. You can create a new currency record at Lists > Currencies.
CRNCY_REQD	currency expected for pricing element
CSC_SETUP_REQD	To display the CSC field on the form, you must enable the "Use Card Security Code for Credit Card Transactions" preference, located on the Setup > Accounting Preferences task.
CSTM_FIELD_KEY_REQD	The specified custom field key is missing.
CSTM_FIELD_VALUE_REQD	The specified custom field value is missing.
CSV_DELIMITER_ERROR	Values in the following CSV file(s) are not comma separated, and cannot be imported. Please reformat the file(s) and try again. For instructions, <a href='javascript:nlPopupHelp("DOC_Reformatting_Semi-Colon_Separated_CSV_Files","help")'>visit this help topic.</a>
CUST_ARLEADY_HAS_ACCT	Attempting to provision a new account to a customer with an existing account
CUST_CNTR_USER_ACCESS_ONLY	This form is only accesible to customer center users.
CUST_LEAD_NOT_GROUP_MEMBR	{1} this customer or lead does not belong to your group.
CYBERSOURCE_ERROR	An unexpected error occurred while processing the credit card through CyberSource (reason code = {1}). Please contact NetSuite support.
CYBERSOURCE_ERROR	: The credit card was declined by the card issuer. Use a different card for payment or contact the card issuer for more information.
CYBERSOURCE_ERROR	The credit card transaction was denied by the issuing bank. Please try another card or contact the card issuer for more information.
CYBERSOURCE_ERROR	The credit card has expired or the expiration date does not match the date on file with the card issuer. Please correct the expiration date or try another card.
CYBERSOURCE_ERROR	The credit card transaction was denied due to insufficient funds. Please try another card or contact the card issuer for more information.
CYBERSOURCE_ERROR	The credit card transaction could not be completed because the issuing bank was not available. Please try another card or wait a few minutes and try again.
CYBERSOURCE_ERROR	Inactive card or card not authorized for card-not-present transactions. Please try another card or contact the card issuer for more information.
CYBERSOURCE_ERROR	The card has reached the credit limit. Please try another card or contact the card issuer for more information.
CYBERSOURCE_ERROR	Invalid card verification number. Please check to make sure you have provided the correct card verification number.
CYBERSOURCE_ERROR	Invalid credit card account number. Please check to make sure you have provided the correct credit card account number.

Error Code Returned	Long Description or Message
CYBERSOURCE_ERROR	The type of credit card provided is not accepted by this merchant. Please try another card or contact the merchant for more information.
CYBERSOURCE_ERROR	Successful transaction.
CYBERSOURCE_ERROR	The request is missing one or more required fields. Possible action: See the reply fields missingField_0...N for which fields are missing. Resend the request with the complete information.
CYBERSOURCE_ERROR	One or more fields in the request contains invalid data. Possible action: See the reply fields invalidField_0...N for which fields are invalid. Resend the request with the correct information.
CYBERSOURCE_ERROR	The merchantReferenceCode sent with this authorization request matches the merchantReferenceCode of another authorization request that you sent in the last 15 minutes. Possible action: Resend the request with a unique merchantReferenceCode value.
CYBERSOURCE_ERROR	Error: General system failure. See the documentation for your CyberSource client (SDK) for information about how to handle retries in the case of system errors.
CYBERSOURCE_ERROR	Error: The request was received but there was a server timeout. This error does not include timeouts between the client and the server. Possible action: To avoid duplicating the order, do not resend the request until you have reviewed the order status in the Business Center. See the documentation for your CyberSource client (SDK) for information about how to handle retries in the case of system errors.
CYBERSOURCE_ERROR	Error: The request was received, but a service did not finish running in time. Possible action: To avoid duplicating the order, do not resend the request until you have reviewed the order status in the Business Center. See the documentation for your CyberSource client (SDK) for information about how to handle retries in the case of system errors.
CYBERSOURCE_ERROR	The issuing bank has questions about the request. You do not receive an authorization code programmatically, but you might receive one verbally by calling the processor. Possible action: Call your processor or the issuing bank to possibly receive a verbal authorization. For contact phone numbers, refer to your merchant bank information.
CYBERSOURCE_ERROR	Expired card. You might also receive this if the expiration date you provided does not match the date the issuing bank has on file. Possible action: Request a different card or other form of payment.
CYBERSOURCE_ERROR	General decline of the card. No other information provided by the issuing bank. Possible action: Request a different card or other form of payment.
CYBERSOURCE_ERROR	Insufficient funds in the account. Possible action: Request a different card or other form of payment.
CYBERSOURCE_ERROR	Stolen or lost card. Possible action: Review the customers information and determine if you want to request a different card from the customer.
CYBERSOURCE_ERROR	Issuing bank unavailable. Possible action: Wait a few minutes and resend the request.
CYBERSOURCE_ERROR	Inactive card or card not authorized for card-not-present transactions. Possible action: Request a different card or other form of payment.
CYBERSOURCE_ERROR	The card has reached the credit limit. Possible action: Request a different card or other form of payment.

Error Code Returned	Long Description or Message
CYBERSOURCE_ERROR	Invalid card verification number. Possible action: Request a different card or other form of payment.
CYBERSOURCE_ERROR	The customer matched an entry on the processors negative file. Possible action: Review the order and contact the payment processor.
CYBERSOURCE_ERROR	Invalid account number. Possible action: Request a different card or other form of payment.
CYBERSOURCE_ERROR	The card type is not accepted by the payment processor. Possible action: Request a different card or other form of payment. Also, check with CyberSource Customer Support to make sure your account is configured correctly.
CYBERSOURCE_ERROR	General decline by the processor. Possible action: Request a different card or other form of payment.
CYBERSOURCE_ERROR	There is a problem with your CyberSource merchant configuration. Possible action: Do not resend the request. Contact Customer Support to correct the configuration problem.
CYBERSOURCE_ERROR	The requested amount exceeds the originally authorized amount. Occurs, for example, if you try to capture an amount larger than the original authorization amount. This reason code only applies if you are processing a capture through the API. See Using the API for Captures and Credits. Possible action: Issue a new authorization and capture request for the new amount.
CYBERSOURCE_ERROR	Processor failure. Possible action: Tell the customer the payment processing system is unavailable temporarily, and to try their order again in a few minutes.
CYBERSOURCE_ERROR	The authorization has already been captured. This reason code only applies if you are processing a capture through the API. See Using the API for Captures and Credits. Possible action: No action required.
CYBERSOURCE_ERROR	The requested transaction amount must match the previous transaction amount. This reason code only applies if you are processing a capture or credit through the API. See Using the API for Captures and Credits. Possible action: Correct the amount and resend the request.
CYBERSOURCE_ERROR	The card type sent is invalid or does not correlate with the credit card number. Possible action: Ask your customer to verify that the card is really the type that they indicated in your Web store, then resend the request.
CYBERSOURCE_ERROR	The request ID is invalid. This reason code only applies when you are processing a capture or credit through the API. See Using the API for Captures and Credits. Possible action: Request a new authorization, and if successful, proceed with the capture.
CYBERSOURCE_ERROR	You requested a capture through the API, but there is no corresponding, unused authorization record. Occurs if there was not a previously successful authorization request or if the previously successful authorization has already been used by another capture request. This reason code only applies when you are processing a capture through the API. See Using the API for Captures and Credits. Possible action: Request a new authorization, and if successful, proceed with the capture.
CYBERSOURCE_ERROR	The capture or credit is not voidable because the capture or credit information has already been submitted to your processor. Or, you requested a void for a type of transaction that cannot be voided. This

Error Code Returned	Long Description or Message
	reason code applies only if you are processing a void through the API. See Using the API for Voids for information about voids. Possible action: No action required.
CYBERSOURCE_ERROR	You requested a credit for a capture that was previously voided. This reason code applies only if you are processing a void through the API. See Using the API for Voids for information about voids. Possible action: No action required.
CYBERSOURCE_ERROR	Error: The request was received, but there was a timeout at the payment processor. Possible action: To avoid duplicating the transaction, do not resend the request until you have reviewed the transaction status in the Business Center.
CYBERSOURCE_ERROR	The authorization request was approved by the issuing bank but declined by CyberSource based on your Smart Authorization settings. Possible action: Do not capture the authorization without further review. Review the ccAuthReply_avsCode, ccAuthReply_cvCode, and ccAuthReply_authFactorCode fields to determine why CyberSource rejected the request.
CYBERSOURCE_ERROR	Unable to process credit card transaction. The code returned from CyberSource {1} is not a recognized reason code. Please contact NetSuite support.
CYCLE_IN_PROJECT_PLAN	The changes made to this entity have cause a cycle in the project plan. Select a different parent and/or predecessors to avoid the cycle.
DASHBOARD_LOCKED	Your dashboard has been set up and locked by an administrator. Please contact them for details.
DATA_MUST_BE_UNIQUE	The update failed because every entry in this column must be unique.
DATA_REQD	You need to provide a proper value for the required field: {1}.
DATA_REQD	You are missing the following required field(s):{1}
DATE_EXPECTED	You entered '{1}' into a field where a calendar date was expected. \nPlease go back and change this value to the correct date.
DATE_PARAM_REQD	missing date parameter
DATE_PRD_MISMATCH	Your transaction date does not fall between the start and end dates of your accounting period.
DEFAULT_CUR_REQD	Default currency cannot be null
DEFAULT_EXPENSE_ACCT_REQD	A default expense account must be specified to activate items on the list.nGo to Setup > Set Up Payroll and click the Default Accounts subtab.nIn the Payroll Expenses Account field, choose a default general ledger account for your payroll expenses. Then, click Save.
DEFAULT_ISSUE_OWNER_REQD	There is no default owner for the issue role {1}. This operation cannot be completed until this is corrected.
DEFAULT_LIAB_ACCT_REQD	A default liability account must be specified to activate items on the list.nGo to Setup > Set Up Payroll and click the Default Accounts subtab.nIn the Payroll Liabilities Account field, choose a default general ledger account for your payroll liabilities. Then, click Save.
DEFAULT_ROLE_REQD	Login Failed because you do not have a default role for the company and email entered. Please Try Again.

Error Code Returned	Long Description or Message
DEFAULT_TYPE_DELETE_ DISALLWD	You cannot delete default types
DEFERRAL_ACCT_REQD	Lines with amortization templates must have a deferral account.
DEFERRAL_ACCT_REQD	Lines with revenue recognition templates must have a deferral account.
DEFERRED_REV_REC_ACCT_ REQD	The {1} item does not have a Deferred Revenue Account specified. Please assign the item a Deferred Revenue Account using the standard User Interface, and then re-import the transaction.
DEPT_IN_USE	Your classes cannot be converted to departments because your existing department records are referred to by transactions or other records. These department records cannot be overwritten.
DFRNT_SWAP_PRICE_ LEVELS_REQD	Please select different price levels to swap prices.
DISALLWD_IP_ADDRESS	The specified IP address rules must allow the login of your current IP Address. Your current IP address is {1}. For information on entering IP address rules, click Help at the top of the page.
DISCOUNT_ACCT_SETUP_ REQD	Please <a href='/app/setup/acctsetup.nl'>Set Up Discount Accounts</a> first.
DISCOUNT_DISALLWD	You have attempted to save this transaction with one or more discounts and where all items have Permit Discount = Never. You must change one of the items to permit a discount, add a new item without the restriction or remove the discount from the transaction.
DISCOUNT_DISALLWD_VSOE	Posting discounts are not allowed on items in VSOE bundles.
DISCOUNT_EXCEED_TOTAL	Discount can not exceed item total.
DISTRIB_REQD_ONE_DAY_ BFORE	All items must be distributed at least one day before they may be transferred.
DOMAIN_IN_USE	The domain {1} is already in use
DOMAIN_WEBSITE_REQD	Please select a Web Site for domain {1}
DROP_SHIP_ERROR	The following error occurred when updating the quantity on the drop ship {1:transaction type}: <p>{2:error message}</p>
DROP_SHIP_ERROR	The transaction was successfully saved, but an error occurred while running user events and email alerts after updating the drop ship {1:transaction type}(s)
DROP_SHIP_ERROR	The transaction was successfully saved, but an error occurred while running user events after updating the drop ship {1:transaction type}(s)
DROP_SHIP_OR_SPECIAL_ ORD_ALLWD	Items can be Drop Ship or Special Order but not both
DUE_DATE_BFORE_START_ DATE	Due date occurs before start date
DUE_DATE_REQD	Please enter a value for {1} Due Date
DUP_ACCT_NAME	The account name you have chosen is already used. Go <a href='javascript:history.go(-1);';>back</a>, change the name and resubmit.
DUP_ACCT_NOT_ALLWD	You may not use duplicate accounts on an item.
DUP_ACCT_NUM	The account number you have chosen is already used.

Error Code Returned	Long Description or Message
DUP_ACCT_NUM	The account number you have chosen is already used. Go <a href="javascript:history.go(-1);";>back</a>, change the number and resubmit.
DUP_ACCT_ON_TRANS	This transaction has duplicate accounts. The main line of the transaction and the line labeled '{1}' both use the account named '{2}'.
DUP_BIN	A bin already exists with that name. Go back, change the name, and resubmit.
DUP_BIN	There is already another bin with that number. Please choose a bin number that is not used by another bin.
DUP_BUNDLE_IN_ACCT	That bundle has already been copied or installed in this account.
DUP_BUNDLE_IN_ACCT	You cannot install this bundle as it is a copy of bundle {1} that you previously installed.
DUP_CATEGORY	This category already exists
DUP_CATEGORY_NAME	A category already exists with that name. Go <a href="javascript:history.go(-1);";>back</a>, change the name and resubmit.
DUP_COLOR_THEME	This color theme already exists
DUP_CSTM_FIELD	This custom field already exists
DUP_CSTM_LAYOUT	This custom layout already exists
DUP_CSTM_LIST	There is already a Custom List or Custom List element with that name
DUP_CSTM_RCRD	There is already a Custom Record with that name
DUP_CSTM_RCRD_ENTRY	There is already a Custom Record Entry with that name
DUP_CSTM_TAB	This custom tab already exists
DUP_EMPL_EMAIL	There is already an employee with external access to this account using that email address. All employees with external access must have a unique email address for login purposes. Go <a href="javascript:history.go(-1);";>back</a>, change the email address and resubmit.
DUP_EMPL_ENTITY_NAME	There is already an employee with external access to this account using that entity name. All employees with external access must have a unique entity name for login purposes. Go <a href="javascript:history.go(-1);";>back</a>, change the entity name and resubmit.
DUP_EMPL_TMPLT	There is already an employee template with that name. Go <a href="javascript:history.go(-1);";>back</a>, change the template name and resubmit.
DUP_ENTITY	This entity already exists.
DUP_ENTITY_EMAIL	There is already an external entity (eg. customer, vendor, or employee) with access to this account using that email address. All external entities with access must have a unique email address for login purposes.
DUP_ENTITY_NAME	There is already an external entity (eg. customer, vendor, or employee) with access to this account using that entity name. All external entities with access must have a unique entity name for login purposes.
DUP_FEDEX_ACCT_NUM	There is an existing NetSuite registration for FedEx account number {1}.

Error Code Returned	Long Description or Message
DUP_FINANCL_STATMNT_ LAYOUT	This financial statement layout already exists.
DUP_INFO_ITEM	This information item already exists
DUP_ISSUE_NAME_OR_NUM	You cannot set {1:issue record name} {2:issue number} to be a duplicate of itself or one of its duplicates.
DUP_ITEM	Uniqueness error - there is already an item with that name or name/ parent combination.
DUP_ITEM_LAYOUT	This item/category layout already exists
DUP_ITEM_NAME	There is already an item with that name. . Go <a href="javascript:history.go(-1);";>back</a>, change the name and resubmit.
DUP_ITEM_OPTION	A child item child with that combination of options already exists
DUP_ITEM_TMPLT	This item/category template already exists
DUP_MATRIX_OPTN_ABBRV	Matrix option '{1}' already uses that abbreviation. Please choose another.
DUP_MEMRZD_TRANS	There is already a Memorized Transaction with that name. Go <a href="javascript:history.go(-1);";>back</a>, change the name and resubmit.
DUP_NAME	That name is already in use. Go <a href="javascript:history.go(-1);";>back</a>, change the name and resubmit.
DUP_PAYROLL_ITEM	There is already a payroll item named {1}
DUP_PRESNTN_CAT	This presentation category already exists
DUP_RCRD	A {1} already exists with that name. Go <a href="javascript:history.go(-1);";>back</a>, change the name and resubmit.
DUP_RCRD	Matched more than one record for {1}
DUP_RCRD	Matched more than one record (internalIds {1} and {2})
DUP_RCRD	This record already exists
DUP_RCRD_LINK	Link to that record already exists
DUP_SALES_TAX_ITEM	You have entered a duplicate Sales Tax Item. Go <a href="javascript:history.go(-1);";>back</a>, change the name, city, state or zip code and resubmit.
DUP_SHIPPING_ITEM	You have entered a duplicate Shipping Item. Go <a href="javascript:history.go(-1);";>back</a>, change the name and resubmit.
DUP_SHORT_NAME	Duplicate short name
DUP_SITE_THEME	This site theme already exists
DUP_SOURCE_ACCT	Duplicate source accounts are not allowed.
DUP_TAX_CODE	You have entered a duplicate Tax Code. Go <a href="javascript:history.go(-1);";>back</a>, change the name and resubmit.

Error Code Returned	Long Description or Message
DUP_TAX_CODE	You have entered a duplicate Tax Code. Go <a href= \"javascript:history.go(-1);\";>back</a>, change the name, city, state or zip code and resubmit.
DUP_TRACKING_NUM	You entered the following tracking number twice: {1}. Note that a single tracking number may not contain spaces or commas. A space or comma will be interpreted as the separator between different tracking numbers. For example, '1029 3847 465' will be interpreted as 3 different tracking numbers. It should be entered without spaces: '10293847465'.
DUP_TRANS	Duplicate Trans. Unable to locate, no match.
DUP_UPS_ACCT_NUM	There is an existing NetSuite registration for UPS account number {1}.
DUP_VENDOR_EMAIL	There is already a vendor with external access to this account using that email address. All vendors with external access must have a unique email address for login purposes. Go <a href='javascript:history.go(-1);';>back</a>, change the email address and resubmit.
DUP_VENDOR_NAME	There is already a vendor using that entity name. All vendors must have a unique entity name. Go <a href="javascript:history.go(-1);";>back</a>, change the entity name and resubmit.
DUPLICATE_INVENTORY_ NUM	Duplicate inventory number found in entry: {1}
DUPLICATE_INVENTORY_ NUM	Duplicate inventory number found on different lines of transaction
DUPLICATE_KEYS	This record contains duplicated key or keys. Please correct it before next update.
DUPLICATE_NAME_FOR_PRD	Please choose a different period name. "{1}" is already taken.
DUPLICATE_NAME_FOR_ ROLE	Please choose a different role name. "{1}" is already taken.
EARNING_ITEM_REQD	At least one employee in this payroll has no earning items. Please make sure that every employee has at least one earning item.
EDITION_DSNT_SUPRT_ WORLDPAY	WorldPay is not supported in this edition.
EIN_OR_TIN_REQD	You must set either the Employer Identification Number (EIN) or SSN/ TIN (Social Security Number, Tax ID Number) to complete this fulfillment. Please go to Setup > Company Information to set either of these fields and re-submit.
EMAIL_ADDRS_REQD	Please enter your email address
EMAIL_ADDRS_REQD_TO_ NOTIFY	Please enter an email address for this company. A notification email will be sent when this case record is saved.
EMAIL_ADDRS_REQD_TO_ NOTIFY	The recipient you are sending this email to does not have an email address. Please enter one and try again
EMAIL_REQ_HANDLER_ ERROR	an error occurred while instantiating an email requesthandler
EMAIL_REQ_HANDLER_ ERROR	An Error occurred while performing system-level validation of email request for <{1}> from <{2}>

Error Code Returned	Long Description or Message
EMAIL_REQ_HANDLER_ERROR	An Error occurred while POSTing data into requestHandlder: {1}
EMAIL_REQ_HANDLER_ERROR	an error occurred while servicing an email requesthandler in state: {1}
EMAIL_REQD	You must enter a valid email address to email the transaction.
EMAIL_REQD_ACCT_PROVISION	Cannot provision an account without an email address for this customer: Was external access granted?
EMPL_IN_USE	You can't delete this employee, as commissions have been calculated for this employee.
EMPL_IN_USE	You can't delete this employee, as it is or has been referenced by other employees as a supervisor.
ERROR_DELETE_CARD_DATA	Failed to delete card data. Retry request.
ERROR_IN_TERRITORY_ASSGNMNT	Error Performing Initial Round_Robin Assignment for Territory: {1}
ERROR_IN_TERRITORY_ASSGNMNT	Error Performing Round_Robin Assignment for Territory: {1}
ERROR_PRCSSNG_TRANS	There were errors processing the selected transactions. Please process them individually for more information.
ERROR_REFUND_TRANS	Failed to refund International transaction. Retry request.
ERROR_REVERSE_AUTH	Failed to reverse International authorization. Retry request.
ERROR_SENDING_TRAN_EMAIL	The transaction was entered successfully, but an unexpected error occurred while sending the transaction email {1}
ERROR_VOID_TRANS	Failed to void International transaction. Retry request.
EVENT_ID_NOT_FOUND	Event ID not found
EXCEEDED_MAX_ALLWD_LOC	You have reached the maximum allowance of {1} location records. If you need to create additional location records, please contact our NetSuite Customer Support team for assistance
EXCEEDED_MAX_CONCUR_RQST	The maximum number of concurrent requests has been exceeded. Please try your request again when an existing session has completed.
EXCEEDED_MAX_EMAILS	This account has {1} more bulk emails that can be sent. If you would like to purchase an additional block of emails, please contact your account manager.
EXCEEDED_MAX_EMAILS	The merge exceeds the number of bulk merge emails allotted to your account this year. This account has {1} more bulk emails that can be sent this year. Please contact your NetSuite account manager to purchase additional block of emails.
EXCEEDED_MAX_EMAILS	This campaign email event exceeds the number of emails ({1}) that can be sent per event without setting up a default campaign domain or specifying one on the campaign email template.
EXCEEDED_MAX_EMAILS	This merge operation exceeds the number of emails ({1}) that can be sent per execution without setting up a bulk merge domain or specifying one on the email template.

Error Code Returned	Long Description or Message
EXCEEDED_MAX_EMAILS	You cannot schedule more than {1} emails per year. If you'd like to purchase an additional block of emails, please contact your account manager.
EXCEEDED_MAX_FEATURED_ ITEMS	There is a limit of {1} featured items on this page.
EXCEEDED_MAX_FIELD_ LENGTH	Address line 1 cannot exceed 35 characters. Please check the shipper and recipient address to ensure the "Address 1" field is a maximum of 35 characters.
EXCEEDED_MAX_FIELD_ LENGTH	Address line 2 cannot exceed 35 characters. Please check the shipper and recipient address to ensure the "Address 2" field is a maximum of 35 characters.
EXCEEDED_MAX_FIELD_ LENGTH	The field {1} contained more than the maximum number ({2}) of characters allowed.
EXCEEDED_MAX_FIELD_ LENGTH	The string "{1}" contained more than the maximum number of characters allowed.
EXCEEDED_MAX_FIELD_ LENGTH	Too many characters for a field
EXCEEDED_MAX_MATRIX_ OPTNS	The total combination of subitems you have selected exceeds the maximum allowed of 2000. Please choose fewer options on the matrix tab.
EXCEEDED_MAX_MATRIX_ OPTNS	The total combination of subitems you have selected exceeds the maximum allowed of 2000. Please choose fewer options on the matrix tab.
EXCEEDED_MAX_PDF_ ELEMENTS	There is a maximum of 100 custom elements allowed on a PDF layout.
EXCEEDED_MAX_PDF_ EXPORT_COL	PDF Export is limited to 30 columns.
EXCEEDED_MAX_PIN_ RETRIES	PIN entered incorrectly too often.
EXCEEDED_MAX_RCRD	You have reached the maximum allowance of {1} {2} records. If you need to create additional {2} records, please contact our NetSuite Customer Support team for assistance
EXCEEDED_MAX_REPORT_ COL	The option you selected in the Column field results in a report that exceeds the maximum number of columns allowed. Please select additional filters or select a shorter date/period range.
EXCEEDED_MAX_REPORT_ ROWS	Reports are limited to {1} rows. Please narrow your results.
EXCEEDED_MAX_REPORT_ SIZE	The results of this report are too large. Please narrow your results.
EXCEEDED_MAX_SESSIONS	Maximum active sessions exceeded. Please wait 5 minutes and login again.
EXCEEDED_MAX_SHIP_ PACKAGE	The maximum number of custom shipping packages has been exceeded: {1}. Please reduce item quantities to generate fewer packages, or enter the packages manually.
EXCEEDED_MAX_TIME	The operation has exceeded maximum allowed time for completion. Operation aborted.

Error Code Returned	Long Description or Message
EXCEEDED_MAX_TRANS_ LINES	Transactions may not contain more than {1} lines.
EXCEEDED_MAX_USERS_ ALLWD	The changes you have made to this employee's access have caused you to exceed either your Full Access or Employee Center allowance. To make these changes you must either adjust the number of employees assigned the Employee Center role or contact your account representative to purchase additional licenses.
EXCEEDED_MAX_USERS_ ALLWD	Assigning this role would exceed your Full Access licenses ({1}). To assign this role, you must remove another employee's full access roles or contact your account representative to purchase additional licenses.
EXCEEDED_MAX_USERS_ ALLWD	The changes you have made to this employee's access have caused you to exceed either your Full Access or Retail User allowance. To make these changes you must either adjust the number of employees assigned the Retail Clerk role or contact your account representative to purchase additional licenses.
EXCEEDED_PER_TRANS_MAX	Exceeded per transaction maximum on account {1}
EXCEEDED_RQST_SIZE_LIMIT	You have exceeded the permitted request size limit ({1})
EXCEEDS_ALLWD_LICENSES	Adding access for this user exceeds the number of licenses you have purchased. To add another user, you must first remove access from an existing user or contact NetSuite to purchase additional licenses.
EXCEEDS_ALLWD_LICENSES	Adding a {1} would exceed the number of licenses you have purchased. Please contact NetSuite for additional licenses.
EXPENSE_ENTRY_DISALLWD	{1} does not allow expense entry.
EXPIRED_SEARCH_CRITERIA	Your search criteria expired. The criteria for a specific search generally expire after 15 minutes of inactivity. Please return to the search definition page and re-submit your search.
EXT_CAT_LINK_SETUP_REQD	Error - you have not properly set up links from your External Catalog Site back into {1}!
EXTERNALID_NOT_ SUPPORTED	Field {1} does not support externalId
EXTERNALID_REQD	This operation requires a value for externalId.
FAILED_FEDEX_LABEL_VOID	Failed FedEx Label Void
FAILED_FORM_VALIDATION	Form validation failed. You cannot submit this record.
FAILED_UPS_LABEL_VOID	Failed UPS Label Void
FAX_NUM_REQD	You must enter a fax number.
FAX_NUM_REQD	You must enter a fax number for this recipient before performing a fax merge operation.
FAX_NUM_REQD	You must enter a valid fax number to fax the transaction.
FAX_SETUP_REQD	Before you can send faxes, you need to go to the <a href='/app/setup/printing.nl'>Set Up Printing, Fax & Email</a> page and set up the fax service.
FEATURE_DISABLED	You do not have the correct features enabled to search on {1}.
FEATURE_DISABLED	The "{1}" feature is not enabled in your {2} account.

Error Code Returned	Long Description or Message
FEATURE_DISABLED	The feature '{1}' required to access this page is not enabled in this account.
FEATURE_UNAVAILABLE	<b>{1} Trial does not allow access to this feature.</b> If you would like more information about this feature, please contact your account manager.
FEATURE_UNAVAILABLE	Error - This business does not have the External Catalog Site feature enabled.
FEATURE_UNAVAILABLE	Test Drive does not allow access to this feature. If you would like more information about this feature, please contact your account manager.
FEATURE_UNAVAILABLE	That feature is only available to Plus users
FEATURE_UNAVAILABLE	The {1} feature is not available to your company.
FEATURE_UNAVAILABLE	This feature is not available to your company.
FED_ID_REQD	Must have Federal Identification Number to process 1099-MISC forms.
FED_WITHHOLDING_REQD	Your employee record does not have current Federal Withholding information.<p>Please contact your supervisor to set up your record with the appropriate information.</p>
FEDEX_ACCT_REQD	The FedEx Account Number has not been set.
FEDEX_CANT_INTEGRATE_FULFILL	The fulfillment cannot be integrated with {1} because the Shipping Integration Carrier is set to UPS.
FEDEX_DROPOFF_TYP_REQD	The FedEx Dropoff Type has not been set.
FEDEX_INVALID_ACCT_NUM	This account number was not recognized by FedEx. Please re-enter your account number, or contact FedEx to open a new account.
FEDEX_ITEM_CONTENTS_REQD	For international shipments, {1} requires specific information about the item contents.
FEDEX_METER_NOT_RETRIEVED	A FedEx Meter Number was not retrieved for account number {1}. Please try your request again in a few minutes.
FEDEX_METER_REQD	The FedEx Meter Number has not been set.
FEDEX_ONE_PACKG_ALLWD	The selected FedEx service allows only one package per fulfillment. If more than one package is required, please break up the shipment into multiple fulfillments of one package each.
FEDEX_ORIGIN_COUNTRY_US_REQD	The origin country must be United States (US) for all Item Fulfillments when using a FedEx shipping method.
FEDEX_RATING_SRVC_UNAVAILBL	The FedEx rating services application is currently unavailable. Please try your request again in a few minutes.
FEDEX_REG_NOT_FOUND	A valid FedEx Registration was not found for the specified location:
FEDEX_SHIP_SRVC_REQD	The FedEx Shipping Service has not been set.
FEDEX_SHIP_SRVC_UNAVAILBL	The FedEx shipping services application is currently unavailable. Please try your request again in a few minutes.
FEDEX_UNSUPRTD_ORIGIN_COUNTRY	The origin country {1} is currently not supported for Item Fulfillments when using a FedEx shipping method.
FEDEX_USD_EXCHANGE_RATE_REQD	Cannot retrieve FedEx realtime rates: USD Exchange Rate is required when requesting FedEx realtime rates.

Error Code Returned	Long Description or Message
FEDEX_VOID_ERROR	The FedEx Void failed due to a system error.
FIELD_CALL_DATE_REQD	Missing Required Field: Call Date
FIELD_DEFN_REQD	Field definition not found
FIELD_NOT_SETTABLE_ON_ ADD	You are not allowed to set the nsKey for a record
FIELD_PARAM_REQD	Please enter a value for {1}
FIELD_PARAM_REQD	Please enter values for {1}.
FIELD_REQD	Mandatory Field Missing
FIELD_REQD	You must first select a field
FILE_ALREADY_EXISTS	A file with the same name already exists in the selected folder.
FILE_ALREADY_EXISTS	Note: You are attempting to upload a file with a name matching an existing file in the selected folder. Please rename this file or select another folder, and then upload your file.
FILE_DISALLWD_IN_ROOT_ FLDR	You attempted to copy a file to the root directory. Only folders can exist in the root directory.
FILE_DISALLWD_IN_ROOT_ FLDR	You attempted to move a file to the root directory. Only folders can exist in the root directory.
FILE_MISSING	File Missing
FILE_NOT_DOWNLOADABLE	Illegal request for a file that isn't downloadable
FILE_NOT_FOUND	File/Media Item {1} not found.
FILE_NOT_FOUND	File not found. Please try your download again.
FILE_REQD	You must enter a file before submitting this form.
FILE_REQD	You must upload a file before creating this media item
FILE_UPLOAD_IN_PROGRESS	Files are currently being uploaded to this account.
FILTER_BY_AMT_REQD	Please enter an amount to filter by.
FINANCE_CHARGE_SET_ PREFS	Finance Charge Item cannot be edited as an item. Please go to the finance charge preferences page to make changes
FINANCE_CHARGE_SETUP_ REQD	Please set <a href='/app/setup/finchargepref.nl'>Finance Charge Preferences</a> first.
FIRST_LAST_NAMES_REQD	Please enter both your first and last name.
FIRST_QTY_BUCKET_MUST_ BE_ZERO	Quantity defined for first quantity bucket must be zero
FLD_VALUE_REQD	Results are incomplete. You must provide a value for field {1}.
FLD_VALUE_TOO_LARGE	Value for field {1} is too large to be processed.
FOLDER_ALREADY_EXISTS	A folder with the same name already exists in the selected folder.
FORM_RESUBMISSION_ REQD	You have logged in to a different user since you navigated to this form. You must re-submit this form as the new user.
FORM_SETUP_REQD	No appropriate forms are enabled for this role. Please contact your Administrator.
FORM_UNAVAILBL_ONLINE	This form is not available online

Error Code Returned	Long Description or Message
FORMULA_ERROR	Your formula has an error in it. It could resolve to the wrong datatype, use an unknown function, or have a syntax error. Please go back, correct the formula, and re-submit.
FRIENDLY_NAME_REQD	Missing Friendly Name. Friendly Name is a required field and it cannot be null or empty.
FULFILL_REQD_FIELDS_ MISSING	For the listed items, please edit the item record and provide values for the specified fields, and retry the fulfillment.
FULFILL_REQD_FIELDS_ MISSING	The {1} field is required to complete this fulfillment. Please return to the International tab on the item fulfillment and provide a value for the specified field and retry the fulfillment.
FULFILL_REQD_PARAMS_ MISSING	Could not perform operation '{1}' since {2} parameter was not set.
FULL_DISTRIB_REQD	You must fully distribute all {1} numbers for {1} numbered items.
FULL_USERS_REQD_TO_ INTEGRATE	Only full {1} users can integrate with partners.
FX_MALFORMED_RESPONSE	Received malformed response from Foreign Exchange source.
FX_RATE_REQD_FEDEX_RATE	Cannot retrieve {1} realtime rates: {2} Exchange Rate is required when requesting {3} realtime rates.
FX_TRANS_DISALLWD	FX transactions not accepted for this account. Contact Merchant e-Solutions.
GETALL_RCRD_TYPE_REQD	The getAll record type is required.
GIFT_CERT_AMT_EXCEED_ AVAILBL	Gift certificate redemption amount exceeds available amount on the gift certificate
GIFT_CERT_AUTH_ALREADY_ EXISTS	Gift certificate authorization code {1} already exists
GIFT_CERT_CAN_BE_USED_ ONCE	A gift certificate may only be used one time on a transaction
GIFT_CERT_CODE_REQD	Gift certificate codes are missing
GIFT_CERT_CODE_REQD	Missing gift certificate authorization code(s). Please go back and enter authorization codes on the {1}.
GIFT_CERT_CODE_REQD	You must specify a gift certificate code.
GIFT_CERT_IN_USE	Another user is using gift certificate {1}
GIFT_CERT_IN_USE	Gift certificate code {1} is already in use
GIFT_CERT_INVALID_NUM	Gift certificate numbers may not contain the '{1}' character.
GROUP_DSNT_EXIST	That group does not exist
GROUP_REQD	You cannot perform a bulk merge operation with an empty group
GROUP_TYPE_REQD	The group type is required.
GRTR_QTY_PRICE_LEVEL_ REQD	Each quantity pricing level must be greater than the previous quantity pricing level.
ILLEGAL_ID	Illegal ID. Please enter a name.
ILLEGAL_PERIOD_ STRUCTURE	Illegal period structure. Date {1} is in multiple periods.

Error Code Returned	Long Description or Message
INACTIVE_CC_PROFILE	The credit card processing profile provided is inactive.
INACTIVE_RCRD_FOR_ROLE	The record for this role has been made inactive.
INAVLID_FILE_TYP	A change has been made to this file's format. You cannot upload this type of file.
INAVLID_FILE_TYP	You attempted to upload a restricted file type. Please try again with a selection from the list below:
INAVLID_ITEM_TYP	Invalid item type [{1}] for item [{2}].
INAVLID_PRICING_MTRX	Invalid Quantity Pricing Matrix for quantity level {1} : Quantity {2}, Base Price {3}
INCOMPATIBLE_ACCT_CHANGE	The account change you have made is incompatible with old transactions. If you need to swap two accounts, you need to do it in 3 steps. For example, to change the income and asset accounts for an item:<ul><li>(1) Change the income account to a temporary account and save</li><li>(2) Change asset account to the old income account and save</li><li>(3) Change the income to the old asset account and save</li></ul>Please contact customer support if you need assistance with this.
INCOMPATIBLE_ACCT_CHANGE	The account change you have made is incompatible with old transactions. Please either change the account selection appropriately or do not request to update past transactions.
INCOMPLETE_BILLING_ADDR	Billing address is incomplete.
INCOMPLETE_FILE_UPLOAD	The upload did not complete correctly. Please try uploading the file again. If you have repeatedly received this error message, please send mail to <a href="mailto:{1}">{2} Technical Support</a>.
INCRCT_ORD_INFO	The order contains incorrect information and was not placed.
INITIALIZE_ARG_REQD	The initialize reference id is required.
INITIALIZE_ARG_REQD	The initialize reference type is required.
INITIALIZE_ARG_REQD	The initialize type is required.
INITIALIZE_AUXREF_REQD	The initialize auxReference type is required.
INSTALL_SCRIPT_ERROR	Installation Script Error
INSUFCNT_NUM_PRDS_FOR_REV_REC	Not enough accounting periods in range specified for revenue recognition.
INSUFCNT_OPEN_PRDS_FOR_REV_REC	Not enough open accounting periods available for revenue recognition.
INSUFFICIENT_CHARS_IN_SEARCH	Global searches must contain at least three characters to prevent excessive matches.
INSUFFICIENT_FLD_PERMISSION	You are attempting to read an unauthorized field: {1}
INSUFFICIENT_FLD_PERMISSION	You cannot access this search because it includes restricted fields. Please contact your administrator.
INSUFFICIENT_FUND	Decline. Insufficient funds.
INSUFFICIENT_PERMISSION	For security reasons, only an administrator is allowed to edit an administrator record.

Error Code Returned	Long Description or Message
INSUFFICIENT_PERMISSION	Global search is not permitted from this role.
INSUFFICIENT_PERMISSION	Insufficient privileges
INSUFFICIENT_PERMISSION	Your issue DB access has been inactivated. Please contact your issue DB administrator.
INSUFFICIENT_PERMISSION	Your current login role does not have an associated Issue Role. Please change to a different role or contact your Issue administrator.
INSUFFICIENT_PERMISSION	Only the owner can make a contact private
INSUFFICIENT_PERMISSION	Only the super user can update or delete bug entries
INSUFFICIENT_PERMISSION	Permission error: you may not edit this role.
INSUFFICIENT_PERMISSION	Permission Violation: partners do not have access to this report.
INSUFFICIENT_PERMISSION	Permission Violation: partners may not delete saved reports.
INSUFFICIENT_PERMISSION	Permission Violation: You cannot delete saved reports not created by yourself.
INSUFFICIENT_PERMISSION	<b>Test Drive does not allow access to this feature.</b> If you would like more information about this feature, please contact your account manager.
INSUFFICIENT_PERMISSION	The restriction settings on your role deny you access to this item.
INSUFFICIENT_PERMISSION	This folder does not exist or you do not have permission to access this folder.
INSUFFICIENT_PERMISSION	This folder does not permit the direct addition of files
INSUFFICIENT_PERMISSION	This order has been partially or fully processed and may not be edited by a user without permission to approve sales orders.
INSUFFICIENT_PERMISSION	User permission level could not be established
INSUFFICIENT_PERMISSION	You do not have permissions to set a value for element {1} due to one of the following reasons: 1) The field is read-only; 2) An associated feature is disabled; 3) The field is available either when a record is created or updated, but not in both cases.
INSUFFICIENT_PERMISSION	Your role does not have permission to provision accounts.
INSUFFICIENT_PERMISSION	You are not allowed to approve your own transactions.
INSUFFICIENT_PERMISSION	You are not authorized to change this event's organizer. Public events may only have their organizer changed by administrators, the event's organizer, or delegates with edit permission to the event's calendar. Private or busy events may only have their organizer changed by the owner.
INSUFFICIENT_PERMISSION	You cannot update a system defined template.
INSUFFICIENT_PERMISSION	You cannot update cases using this form.
INSUFFICIENT_PERMISSION	You can not access this page unless you are logged in as the consolidated parent company.
INSUFFICIENT_PERMISSION	You can only delete notes that you created.
INSUFFICIENT_PERMISSION	You do not have access to the activity history for that record
INSUFFICIENT_PERMISSION	You do not have access to the media item you selected.

Error Code Returned	Long Description or Message
INSUFFICIENT_PERMISSION	You do not have access to this page
INSUFFICIENT_PERMISSION	You do not have access to this template
INSUFFICIENT_PERMISSION	You do not have permission to access this list.
INSUFFICIENT_PERMISSION	You do not have permission to access this register.
INSUFFICIENT_PERMISSION	You do not have permission to access this type of transaction.
INSUFFICIENT_PERMISSION	You do not have permission to create this type of record. Please choose a different record type.
INSUFFICIENT_PERMISSION	You do not have permission to email transactions.
INSUFFICIENT_PERMISSION	You do not have permission to perform this operation.
INSUFFICIENT_PERMISSION	You do not have permission to print {1}
INSUFFICIENT_PERMISSION	You do not have permission to view this page.
INSUFFICIENT_PERMISSION	You do not have privileges to approve commissions.
INSUFFICIENT_PERMISSION	You do not have privileges to create commissions.
INSUFFICIENT_PERMISSION	You do not have privileges to create this transaction.
INSUFFICIENT_PERMISSION	You do not have privileges to perform that operation.
INSUFFICIENT_PERMISSION	You do not have privileges to perform this action.
INSUFFICIENT_PERMISSION	You do not have privileges to perform this operation
INSUFFICIENT_PERMISSION	You do not have privileges to use this page.
INSUFFICIENT_PERMISSION	You do not have privileges to view this account
INSUFFICIENT_PERMISSION	You do not have privileges to view this page
INSUFFICIENT_PERMISSION	You may not create a new Liability Adjustment or edit existing Liability Adjustments.
INSUFFICIENT_PERMISSION	You may not delete built-in audiences.
INSUFFICIENT_PERMISSION	You may not delete built-in categories.
INSUFFICIENT_PERMISSION	You may not delete built-in items.
INSUFFICIENT_PERMISSION	You may not delete built-in tabs.
INSUFFICIENT_PERMISSION	You must have either 'Transactions -> Invoice' or 'Transactions -> Cash Sale' permission to bill sales orders.
INSUFFICIENT_PERMISSION	You must have either 'Transactions -> Invoice' or 'Transactions -> Cash Sale' permission to fulfill sales orders.
INSUFFICIENT_PERMISSION	You must have 'Transactions -> {1}'permission to build work orders.
INSUFFICIENT_PERMISSION	You must have 'Transactions -> Fulfill Sales Orders' view permission to view sales order fulfillments.
INSUFFICIENT_PERMISSION	You must have 'Transactions -> Fulfill Sales Orders' edit permission to fulfill sales orders.
INSUFFICIENT_PERMISSION	You need employee access to delete this record.
INSUFFICIENT_PERMISSION	{1} The {2} restrictions on your role deny you access to this record.
INSUFFICIENT_PERMISSION	{1} The {2} restrictions on your role prevent you from seeing this record.

Error Code Returned	Long Description or Message
INSUFFICIENT_PERMISSION	{1} The customer restrictions on your partner role prevent you from seeing this record.
INSUFFICIENT_PERMISSION	{1} The restrictions on your role deny you access to this record.
INSUFFICIENT_PERMISSION	{1} The restrictions on your role do not allow you to modify this record.
INSUFFICIENT_PERMISSION	{1} You need {2} the '{3}' permission to access this page. Please contact your account administrator.
INSUFFICIENT_PERMISSION	{1} You need a higher level of the '{2}' permission to access this page. Please contact your account administrator.
INSUFFICIENT_PERMISSION	{1} You need a higher permission for custom record type {2} to access this page. Please contact your account administrator.
INTEGER_REQD_FOR_QTY	Quantity must be an integer for numbered items.
INTL_FEDEX_ONE_PACKG_ALLWD	International FedEx fulfillments allow only one package. If more than one package is required, please break up the shipment into multiple fulfillments of one package each.
INTL_SHIP_EXCEED_MAX_ITEM	The maximum number of items for FedEx International shipping has been exceeded: {1}
INVALID_ABN	Invalid ABN registration number {1}.
INVALID_ACCT	Invalid login. No such account.
INVALID_ACCT	Invalid account number.
INVALID_ACCT_NUM_CSTM_FIELD	The account number custom field does not exist!! Consult billing cell.
INVALID_ACCT_PRD	You can not create an accounting period that is not a year or does not belong to a year.
INVALID_ACCT_TYP	Invalid account type [{1}].
INVALID_ACCT_TYP	There is no account of type: {1}
INVALID_ACCT_TYP	The account and its parent have different account type.
INVALID_ACCT_TYP	You cannot change an account to or from A/R or A/P
INVALID_ACTION	You have attempted an unsupported action.
INVALID_ADDRESS_OR_SHIPPER_NO	An error has occurred. Please ensure that the address information and shipper number are correct, then resubmit the form.
INVALID_ADJUSTMENT_ACCT	The account you selected in Adjustment Account is the same as the asset account for one of the items you are adjusting. Please go back and change the account. Normally, the adjustment account would be an expense account.
INVALID_AES_FTSR_EXEMPTN_NUM	The AES/FTSR Exemption Number is invalid.
INVALID_ALLOCTN_METHOD	You have attempted to allocate landed costs to a transaction using an allocation method that results in no allocation for any lines in the transaction. The allocation method you chose is {1}. To correct this problem, go back to the transaction and choose a different allocation method, or modify the items/lines on the transaction so that there will be some cost allocated to the lines.

Error Code Returned	Long Description or Message
INVALID_AMORTZN_ACCT	The destination account for the amortization schedule could not be determined.
INVALID_AMT	Amount applied greater than total payments and credits
INVALID_AMT	Foreign currency transactions that use Revenue Commitments cannot have a non-discount line with a negative amount. Please use a non-posting discount item instead.
INVALID_AMT	Amount Error (credit) Insufficient Funds (debit). Transaction amount is 0 or too long. Re-submit transaction with a valid amount.
INVALID_AMT	No Action Taken (credit) PIN entry necessary (debit). Reversal amount larger then original amount. No action to take, transaction may not qualify for best level of Interchange.
INVALID_AMT	Base and consumer amounts are inconsistent with the FX rate. Correct provided amounts.
INVALID_APP_ID	Invalid application id: {1}
INVALID_ASSIGN_STATUS_COMBO	Invalid assignee/status combination({1}/{2})
INVALID_ASSIGN_STATUS_COMBO	Invalid assignee/status combination (assignee {1}, status {2}, issue #{3}). No default owner for issue role?
INVALID_ASSIGN_STATUS_COMBO	Invalid assignee/status combination ({1},{2})
INVALID_AUTH	Invalid double authorization. The order is currently authorized for {1}, valid until {2}.
INVALID_AUTH_CODE	You have entered an invalid authorization code for this campaign email address. Please check the authorization code in the email message, and enter it again.
INVALID_AUTOAPPLY_VALUE	Ambiguous data: <autoApply> has been selected and lines have been selected in the <applyList> element.
INVALID_AVS_ADDR	AVS Address Length Error. Please correct the AVS Address and re-submit.
INVALID_AVS_ADDR	Invalid AVS address or zip data. Please correct the AVS data and re-submit.
INVALID_AVS_ZIP	AVS Zip Length Error. Please correct the AVS Zip and re-submit.
INVALID_BALANCE_RANGE	Your balance is not within the allowed range.
INVALID_BILLING_SCHDUL	The billing schedule definition is incompatible with this transaction. Please modify the current billing schedule or select a different one.
INVALID_BILLING_SCHDUL_DATE	Billing schedules may not extend beyond 500 years from today
INVALID_BILLING_SCHDUL_ENTRY	You cannot create a billing schedule with two entries on the same date. Please go back and edit the billing schedule or start date.
INVALID_BIN_NUM	Bin numbers may not contain the '{1}' character
INVALID_BOM_QTY	Inventory/Assembly quantities cannot be negative
INVALID_BOOLEAN_VALUE	Checkbox / boolean data must be either 'T' or 'F'
INVALID_BUG_NUM	Bug number specified was incorrect. ("{1}" isn't a number.)

Error Code Returned	Long Description or Message
INVALID_CAMPAIGN_ CHANNEL	You cannot use this channel to setup this event
INVALID_CAMPAIGN_ GROUP_SIZE	While in {1}, you can only send {2} emails per campaign event. Please modify one or more of your target groups to contain {2} members or less. All campaign emails will be sent to your {1} login email address.
INVALID_CAMPAIGN_STATUS	You cannot set the status of this campaign event back to 'In Progress' because it already has some activity.
INVALID_CARD	Missing Card Holder Account Data. Please provide valid card data and re-submit.
INVALID_CARD	Card not in authorizers database.
INVALID_CARD	The referencing transaction (i.e. reversal request) was not carried out with the same card as the original transaction. Re-submit with original card.
INVALID_CARD_ID	Invalid Card ID. Provide a valid card ID.
INVALID_CARD_NUM	Invalid Card Number. Please provide a valid card number and re-submit.
INVALID_CARD_NUM	Card No. Error. Card number has unknown Bank Identification Number (BIN) or fails check digit edit. Re-try card number again, verify with merchant it is a valid card with proper logos for card types the merchants accept.
INVALID_CARD_TYP	Card Type Not Accepted. Contact Merchant e-Solutions to add the card type.
INVALID_CARD_TYP	Merchant doesn't accept the transaction's card type.
INVALID_CASE_FORM	You cannot create cases using this form.
INVALID_CATGRY_TAX_ AGENCY_REQ	A Vendor must be created in a cateogry with the Tax Agency checkbox checked.
INVALID_CC_EMAIL_ ADDRESS	The email address to CC store orders to is invalid. Please go back and correct it.
INVALID_CC_NUM	Credit card numbers must contain between 13 and 20 digits.
INVALID_CC_NUM	Credit card number is not valid. Please check that all digits were entered correctly.
INVALID_CC_NUM	Credit card number must contain only digits.
INVALID_CERT	Intercompany Transfer Orders cannot be entered when the Accounting Preferences > Order Management > Use Item Cost as Transfer Cost preference is enabled.
INVALID_CERT	Invalid CA certificate
INVALID_CERT	Invalid certificate key
INVALID_CERT	Invalid certificate
INVALID_CERT	Failed to verify client certificate, send email to {1} for help.
INVALID_CERT_AUTH	The indicated CA is not the issuer of this certificate
INVALID_CHARGE_AMT	Charge amount too large. This order can't be charged for more than {1}
INVALID_CHARS_IN_EMAIL	Email address contains invalid characters.

Error Code Returned	Long Description or Message
INVALID_CHARS_IN_NAME	The From Name field cannot contain apostrophes, quotation marks, commas, or greater than or less than signs.
INVALID_CHARS_IN_NAME	You cannot use the colon ':' character in the topic name - please remove it .
INVALID_CHARS_IN_PARAM_FIELD	The Additional Parameters field can not contain any of the following characters: "?\<>|/!@#$%^*()+,.:;'"". Please remove them and try again
INVALID_CHARS_IN_URL	Spaces are not allowed in the {1}url.<p>Examples of a valid {1}url are: <b>http://www.mydomain.com/image.gif</b> or <b>https://one.two.org/user-name/test.jpg</b>
INVALID_CHARS_IN_URL	The URL component you have chosen contains a space or one of the following prohibited character: &?\<>|/!@#$%^&*()+=,.:;'". Please remove them and try again
INVALID_CHECKOUT_EMAIL	The email address to email checkout errors to is invalid. Please go back and correct it.
INVALID_CITY	Merchant City Length Error. Reduce the city name length.
INVALID_COLUMN_NAME	Invalid column name in get_invtitem_col_sum_all_locs: {1} [{2}]
INVALID_COLUMN_VALUE	An attempt was made to set a column to an invalid value. Dynamic SQL being executed [{1}]
INVALID_CONTENT_TYPE	Invalid content type. You can only use application/json, application/xml or text/plain with RESTlets.
INVALID_COSTING_METHOD	SERIAL and LOT are the only costing methods that may be passed as parameters to this page.
INVALID_CRNCY_EXCH_RATE	Invalid currency conversion rate.
INVALID_CRYPT_KEY	{1} is not a valid cryptographic key written as a hexadecimal number.
INVALID_CSTM_FIELD_DATA_TYP	The customfield [{1}] reference object does not match its data type.
INVALID_CSTM_FIELD_RCRD_TYP	Invalid custom field record type
INVALID_CSTM_FIELD_REF	The specified custom field reference {1} is invalid.
INVALID_CSTM_FORM	{1} is an invalid custom form
INVALID_CSTM_RCRD_KEY	Invalid custom record key [{1}].
INVALID_CSTM_RCRD_QUERY	Invalid custom record object in query.
INVALID_CSTM_RCRD_TYPE_KEY	Invalid custom record type
INVALID_CSTM_RCRD_TYPE_KEY	Invalid custom record type key.
INVALID_CSTM_RCRD_TYPE_KEY	{1} refers to a custom list. To get the contents of this list, use the 'get' or 'getAll' operation with a RecordRef of type 'customList'
INVALID_CSTM_RCRD_TYP_KEY	Invalid custom record type key in query.
INVALID_CUR	You have entered an invalid currency symbol or internal ID: {1}.

Error Code Returned	Long Description or Message
INVALID_CURR_CODE	Failed to find a currency code for the currency symbol {1}. Verify your currency symbol is ISO-compliant and re-submit.
INVALID_CURRENCY_CODE	Failed to find currency code for the requested country code. Check country code and retry request.
INVALID_CURRENCY_CODE	Request currency code must match FX rate currency code. Retry request with a currency code that matches the FX currency code.
INVALID_CURRENCY_TYP	Currency Type Not Accepted. Contact Merchant e-Solutions to add the currency type.
INVALID_CUSTOMER_RCRD	This customer record {1} is not valid. Please create the customer first.
INVALID_DATA	Invalid data combination, can not set {1} to {2} and {3} to {4}
INVALID_DATA_FORMAT	Invalid data format. You should return a JavaScript object.
INVALID_DATA_FORMAT	Invalid data format. You should return TEXT.
INVALID_DATE	Date Error. Invalid date.
INVALID_DATE	The date < {1} > is invalid. You must specify a date after < {2} >.
INVALID_DATE_FORMAT	Date field not in your preferred date format
INVALID_DATE_RANGE	The date range you specified does not enclose all its child periods.
INVALID_DATE_RANGE	Invalid time range. The {1} "{2}" start time ({3}) must be earlier than its end time ({4}).
INVALID_DATE_RANGE	Invalid Date Range - the To Date value must be on or after the From Date value
INVALID_DATE_RANGE	The date range you specified does not fall inside that of the parent period.
INVALID_DEAL_RANGE	Invalid Deal Range - low must be less than projected and high must be greater than projected.
INVALID_DEAL_RANGE	Invalid Deal Range - low must be less than projected and high must be greater than projected.
INVALID_DELETE_REF	Either RecordRef or CustomRecordRef should be used for 'delete' operation.
INVALID_DESTINATION_FLDR	The destination folder is the same as the current folder.
INVALID_DESTNTN_COUNTRY	The destination Country is invalid or has not been set.
INVALID_DESTNTN_POST_CODE	The destination Postal Code is invalid or has not been set.
INVALID_DESTNTN_STATE	The destination State is invalid or has not been set.
INVALID_DETACH_RECORD_TYP	Missing or Invalid RecordType for DetachFrom.
INVALID_DETACH_RECORD_TYP	Detaching of record type {1} from {2} is not supported.
INVLAID_DISCOUNT_MARKUP	Posting and non-posting discounts/markups are not allowed on the same transaction
INVALID_DOMAIN_KEY	The private domain key is invalid, please enter a valid private domain key.

Error Code Returned	Long Description or Message
INVALID_DOMAIN_NAME	Invalid domain name {1}, please enter a valid domain name.
INVALID_DUP_ISSUE_REF	Cannot set this issue to be a duplicate of itself or of an issue that is a duplicate of this issue.
INVALID_EMAIL	Email address is not valid.
INVALID_EMAIL	Your email or code is invalid. Please try again
INVALID_EMAIL	You have entered an invalid email address. Please try again.
INVALID_EMAIL_ADDR	Some of the email addresses you have entered are invalid: {1:list of invalid email addresses}
INVALID_EMAIL_ADDR	The email address for the web store is invalid. Please go back and correct it.
INVALID_END_DATE	You entered an end date ({1}) that is before the start date ({2})
INVALID_END_DATE	{1} [{2}] recurrence end date is invalid
INVALID_END_TIME	invalid 'end' time
INVALID_ENTITY_ INTERNALID	Attempt to insert entity with nkey -1 or 0
INVALID_ENTITY_STATUS	You entered an invalid entity status.
INVALID_EVENT_TIME	You cannot make the time that close to the start or end of the day, because it shifts the event across a day boundary.
INVALID_EXP_DATE	Invalid expiration date. Please correct the expiration date and re-submit.
INVALID_EXPNS_ACCT_SUB	The expense account associated with expense category '{1:expense category}' is not available in the subsidiary of customer '{2:customer}'.
INVALID_EXPRESSION	ERROR: Invalid Expression
INVALID_FAX_NUM	The Fax Number is invalid.
INVALID_FAX_PHONE_ FORMAT	Invalid FaxPhoneNumber. The format of FaxPhoneNumber must contain area code plus seven digit number.
INVALID_FIELD_FOR_RCRD_ TYP	Record type {1} does not support field {2}
INVALID_FIELD_NAME_FOR_ NULL	The specified name [{1}] must exactly match an existing field name.
INVALID_FILE	Verify that you have a valid file to upload.
INVALID_FILE_ENCODING	The file encoding: {1} is not valid. Please refer to the documentation for a list of supported file encodings.
INVALID_FILE_TYP	Invalid file type. File is not a compressed/zip file.
INVALID_FILE_TYP	Invalid file type. File is not a compressed zip file.
INVALID_FILE_TYP	The media file type you uploaded was not recognized. Please try again.
INVALID_FLD	ERROR: Field Not Found
INVALID_FLD_RANGE	Value outside of valid min/max range for this field
INVALID_FLD_TYPE	Application error: NLField of type {1} is not supported.
INVALID_FLD_VALUE	You have entered an Invalid Field Value {1} for the following field: {2}
INVALID_FLDR_SIZE	Error in update_folder_size

Error Code Returned	Long Description or Message
INVALID_FORMAT_IN_ PARAM_FIELD	The Additional Parameters field is not formatted correctly. Please reformat and try again
INVALID_FORMULA	Your formula contains a reference to an encrypted field. This is not allowed.
INVALID_FORMULA	Your formula could result in a divide by zero error. Please go back, correct the formula and resumbit.
INVALID_FORMULA_FIELD	Your formula has an unrecognized field in it. Please go back and correct the formula and resubmit.
INVALID_FROM_DATE	invalid 'from' date
INVALID_FROM_TIME	invalid 'from' time
INVALID_FULFILMNT_ITEM	You have an invalid item {1} in the fulfillment request.
INVALID_FX_BASE_ CURRENCY	FX amount in base currency is required. Provide the base amount.
INVALID_FX_RATE	Exchange Rate must be 1 for vendors in your currency.
INVALID_GET_REF	Either RecordRef or CustomRecordRef should be used for 'get' operation.
INVALID_GIFT_CERT	Invalid gift certificate
INVALID_GIFT_CERT_AMT	The remaining amount on a gift certificate can not be negative
INVALID_GIFT_CERT_CODE	Gift certificate code must contain only letters and digits.
INVALID_GROUP_TYP	This type of group cannot be defined based on another group of the same type.
INVALID_GROUP_TYP	You cannot define this group type using this search.
INVALID_GROUP_TYPE	The group type {1} is invalid.
INVALID_GRP	This type of group cannot be defined based on another group.
INVALID_GST_PST_AGENCIES	The GST or PST agencies are not valid. Please review your company preferences
INVALID_ID	No order found with id {1}
INVALID_ID	Identifiers can contain only digits, alphabetic characters, or "_" with no spaces
INVALID_ID	You have provided an invalid script id or internal id: {1}
INVALID_ID	The externalId attribute is not supported for {1}
INVALID_INITIALIZE_ARG	The reference type {1} and initialize type {2} are not matched.
INVALID_INITIALIZE_ARG	InitializeRef should be used for 'initialize' operation.
INVALID_INITIALIZE_AUXREF	Invalid initialize operation argument 'auxReference'.
INVALID_INITIALIZE_REF	You can not initialize {1}: invalid reference {2}.
INVALID_INITIALIZE_REF	Can not initialize customerPayment: invalid customer reference {1}.
INVALID_INITIALIZE_REF	Can not initialize customerPayment: invalid invoice reference {1}.
INVALID_INITIALIZE_REF	You have an invalid sales order {1} or the order is already billed
INVALID_INITIALIZE_REF	You have an invalid sales order {1} or the order is already closed.

Error Code Returned	Long Description or Message
INVALID_INSURED_VALUE	The Insured Value cannot exceed the total sum of the items being shipped.
INVALID_INTERNAL_ID	The specified internal id is not allowed.
INVALID_INTERNALID	Unparseable Internal Id, did you mean to lookup this field by Name or External ID?
INVALID_INV_DATE	Invoice date on billing schedule may not be after {1}
INVALID_INVENTORY_NUM	Invalid set of inventory numbers: values must be separated by commas, spaces, tabs, or line feeds.
INVALID_INV_DATE	Invoice date on billing schedule may not be after {1}
INVALID_IP_ADDRESS_RULE	The following IP Address rule is not valid: {1}
INVALID_ISSUE_BUILD_ VERSION	Cannot set issue {1} to {2} {3} and {4} {5} because that version is not associated with that build.
INVALID_ISSUE_PRIORITY	Severity 1 issues must have priority 1.
INVALID_ISSUE_PRODUCT	Cannot set issue {1} to {2} {3} and {4} {5} because that product is not associated with that build.
INVALID_ISSUE_PRODUCT	Cannot set issue {1} to {2} {3} and {4} {5} because that product is not associated with that module.
INVALID_ISSUE_STATUS	Cannot set issue {1} to status {2} and assignee {3} because that status requires an assignee with issue role {4}.
INVALID_ITEM_OPTION	Invalid item option {1} for item {2}
INVALID_ITEM_OPTIONS	The options for item '{1}' are no longer available. Please change your order and try again.
INVALID_ITEM_SUBTYP	Invalid item subtype [{1}] for item [{2}].
INVALID_ITEM_TYP	The item [{1}] does not have a valid item type.
INVALID_ITEM_WEIGHT	The total item weight must be > 0.0
INVALID_JOB_ID	You have specified an invalid Job Id
INVALID_KEY_OR_REF	The specified key is invalid.
INVALID_KEY_OR_REF	Invalid {1} reference key {2}.
INVALID_KEY_OR_REF	Invalid {1} reference key {2} for {3} {4}.
INVALID_KEY_PASSWORD_ REQD	This key is invalid or may require a password
INVALID_LINE_ID	No line items match the entered id(s) {1}.
INVALID_LINK_SUM	Links sum to more than applied transaction amount
INVALID_LINK_SUM	Links sum to more than original transaction amount
INVALID_LIST_ID	You must specify a valid line ID. Please set {1}.
INVALID_LIST_KEY	Could not perform operation "{1}" on an invalid line [{2}].
INVALID_LIST_KEY	Could not perform operation 'add' on an existing line [{1}].
INVALID_LOC	Item Fulfillment/Item Receipt location does not match the location on the Transfer Order

Error Code Returned	Long Description or Message
INVALID_LOC_SUB_ RESTRICTN	You may not add inventory to a location that is incompatible with the subsidiary restrictions for this item.
INVALID_LOGIN	Invalid login. Online Form access is disabled.
INVALID_LOGIN	Invalid login. Supplier access is disabled.
INVALID_LOGIN_ATTEMPT	Invalid login attempt.
INVALID_LOGIN_ CREDENTIALS	A problem occured verifying the presented email address, password, roleName or account number, please verify these pieces of information and try again
INVALID_LOGIN_ CREDENTIALS	You have entered an invalid email address or account number. Please try again.
INVALID_LOGIN_ CREDENTIALS	You have entered an invalid email address or password. Please try again.
INVALID_LOGIN_ CREDENTIALS	You have entered an invalid login password. Please try again.
INVALID_LOGIN_ CREDENTIALS	You have entered an invalid password. Please try again.
INVALID_LOGIN_IP	Invalid login. IP Address does not match any of the IP Address rules specified for this entity.
INVALID_LOT_NUM_FORMAT	Lot numbers must be entered using this format: LOT#(Quantity).nFor example, to enter a quantity of 100 items as Lot number ABC1234, enter "ABC1234(100)" in the Lot Numbers field.
INVALID_ MACRO_ID	The Macro ID: {1} is not valid! Please refer to the documentation for a list of supported macro IDs.
INVALID_MARKUP_ DISCOUNT	Markup/Discount % must be between -999% and 999%
INVALID_MCC	Merchant Category Code Length Error. Provide a valid MCC.
INVALID_MEMBER_ HIERARCHY	You have defined a group/kit/assembly item that contains a loop in the member hierarchy. You must remove any group/kit/assembly member items that contain this item as a member.
INVALID_MEMRZD_TRANS	A memorized transaction may not contain any serial or lot numbers. Go back, remove the numbers, and try to re-Memorize the transaction. Posting transactions such as Bills or Cash Sales may not use serial or lot numbered items. Non-Posting transactions such as Purchase Orders or Sales Orders may use serial or lot numbered items but may not contain serial or lot numbers.
INVALID_MERCHANT_KEY	Merchant key is not supplied or incorrect.
INVALID_MERCHANT_NAME	Merchant Name Length Error. Reduce the merchant name length.
INVALID_NAME	Invalid savepoint name. Must start with an alphabet character and can only contain alphanumeric, underscore, dollar, and hash characters.
INVALID_NEXUS	Transaction Nexus is incorrect: it is {1} but should be {2}
INVALID_NUM	Invalid Decimal Number
INVALID_NUMBER	Invalid Decimal Number
INVALID_NUMBER	Invalid Integer

Error Code Returned	Long Description or Message
INVALID_NUMBER	Invalid integer {1}
INVALID_NUMBER	Invalid number {1}
INVALID_NUMBER	You entered "{1}" into a field where a numeric value was expected. Please go back and change this value to a number.
INVALID_NUMBER	You entered an invalid number: Go <a href="javascript:history.go(-1);";>back</a>, change this value and resubmit.
INVALID_OBJ	There are no objects of this type
INVALID_ONLINE_FORM	Online Form not found
INVALID_ONLINE_FORM	This online form is inactive or not available online.
INVALID_ONLINE_FORM_URL	You cannot submit Online forms from this URL. Use the live version: {1} instead
INVALID_OPENID_DOMAIN	This is not a valid domain. Please go back and enter your domain name without prefixes such as 'http://' or 'www'.
INVALID_OPERATION	That operation is not supported for this record type: {1}
INVALID_ORD_STATUS	This order has been partially or fully processed and may not be reset to 'Pending Approval'.
INVALID_ORIGIN_COUNTRY	The origin Country is invalid or has not been set.
INVALID_ORIGIN_POSTCODE	The origin Postal Code is invalid or has not been set.
INVALID_ORIGIN_STATE	The origin State is invalid or has not been set.
INVALID_PAGE_INDEX	Job {1} does not have a page {2}
INVALID_PAGE_PARAM	Invalid page parameter. Unable to view page.
INVALID_PAGER_NUM	The Pager Number is invalid.
INVALID_PARAM	Please select either {1} or {2} parameter but not both.
INVALID_PARENT	An account cannot be its own parent
INVALID_PARTNER_CODE	An account for this customer cannot be provisioned unless its partnercode ({1}) is empty or numeric.
INVALID_PARTNER_ID	Invalid partner id: {1}
INVALID_PASSWORD	Invalid key password
INVALID_PAYCHECK_DATE	Paychecks for {1} must be on or after {2}
INVALID_PERIOD	A period may be only an adjustment period, a quarter, or a year.
INVALID_PHONE	The phone number of the {1} address is invalid. Please verify the phone number is correctly formatted and includes the area code.
INVALID_PHONE_FAX_PAGER_NUM	The Phone, Fax, or Pager Number is invalid.
INVALID_PHONE_NUM	The Phone Number is invalid.
INVALID_PICKUP_POSTAL_CODE	An error has occurred. Pickup Postal Code {1} is not the postal code associated with Shipper Number {2}.
INVALID_PIN	Incorrect PIN. PIN number may have been entered incorrectly. Re-submit with proper PIN.

Error Code Returned	Long Description or Message
INVALID_PIN_DEBIT_TRANS_TYP	Invalid pin debit transaction type. Only a sale (D) is supported.
INVALID_PORTLET_TYP	unsupported portlet type [{1}], id [{2}] processed by cardMetaDataGenerator
INVALID_POST	Invalid Post
INVALID_PRESENTATION_TYP	Presentation Type not recognized
INVALID_PROBABILITY_RANGE	Probability must be between 0 and 100.
INVALID_PROFILE_ID	Invalid Profile ID or Profile Key. Correct the profile ID and profile key, and re-submit.
INVALID_PROJ_BILLING_TYP	The project billing type is incompatible with the billing schedule on the transaction. Please select a different billing schedule.
INVALID_PST_TAX_VALUE	PST tax value is not a valid number: {1}
INVALID_PSWD	Email address "{1}" has been previously registered under a different password from the new password you just provided. For security reasons, you will first need to go back and supply the correct new password for "{1}" to merge the accounts.
INVALID_PSWD	Invalid Password. The password must be between 6 and 10 character with at least one numeric and one alphabetic character.
INVALID_PSWD	Password must be at least 6 characters long.
INVALID_PSWD	Password must be at least 6 characters long and contain at least one number or special character.
INVALID_PSWD	Password must contain at least one letter (A-Z).
INVALID_PSWD	Password must contain at least one number or special character.
INVALID_PSWD	The current password you supplied is incorrect.
INVALID_PSWD	Your new password must be at least {1} characters, contain at least one non-letter, and be substantially different from the current password.
INVALID_PSWD	Your new password must be at least 6 characters, contain at least one non-letter, and be substantially different from the current password.
INVALID_PSWD	Your password cannot be the same as your login. Please choose a new password.
INVALID_PSWD	Your password must be at least 6 characters
INVALID_PSWD	You've used that password before. Please choose a new password.
INVALID_PSWD_HINT	Your hint is too similar to your password. Please choose something less obvious.
INVALID_PSWD_ILLEGAL_CHAR	Password contains an illegal character.
INVALID_PURCHASE_TAX_CODE	Purchase tax code not defined properly for item
INVALID_QTY	The new quantity and new value must be either both positive or both negative.

Error Code Returned	Long Description or Message
INVALID_QTY	You may not receive a larger quantity than you shipped.
INVALID_QUANTITY	Serial and lot number quantities must be integers
INVALID_QUANTITY	Serial and lot number quantities must be positive.
INVALID_QUESTION	Please select a different question.
INVALID_QUESTION	Please select a question.
INVALID_RCRD	Invalid record specification: {1}
INVALID_RCRD_ CONVERSION	Only customer records can be converted to child or parent records. Please select only customer records for this duplicate merge operation.
INVALID_RCRD_HEADER_	Invalid record header: Unable to parse field name from {1}
INVALID_RCRD_HEADER_	Invalid record header: Unable to parse record id from {1}
INVALID_RCRD_HEADER_	Invalid record header: Unable to parse record type from {1}
INVALID_RCRD_ID	Invalid id {1} to create a record.
INVALID_RCRD_INITIALIZE	You have entered an invalid default value for this record initialize operation.
INVALID_RCRD_OBJ	You do not have a valid record object.
INVALID_RCRD_REF	Invalid RecordRef internalId {1} for field {2}
INVALID_RCRD_REF	Invalid record reference.
INVALID_RCRD_REF	Invalid record reference
INVALID_RCRD_TRANSFRM	You have entered an invalid default value for this record transformation operation.
INVALID_RCRD_TRANSFRM	That type of record transformation is not allowed. Please see the documentation for a list of supported transformation types
INVALID_RCRD_TRANSFRM	That is not a valid record transformation.
INVALID_RCRD_TYPE	Invalid Record Type
INVALID_RCRD_TYPE	{1}: type argument {2} is not a valid record or is not available in your account. Please see the documentation for a list of supported record types.
INVALID_RCRD_TYPE	The record type [{1}] is invalid.
INVALID_RCRD_TYPE	The record type is invalid.
INVALID_RECIPIENT	Recipient internal id does not match an existing entity.
INVALID_RECR_REF	Could not update {1} to {2} because referenced record does not exist
INVALID_RECUR_DATE_ RANGE	This event recurrence is invalid because its duration is either negative or longer than one day. {1}
INVALID_RECUR_DATE_ RANGE	This event recurrence is invalid because its end-by date is before its start date. {1}
INVALID_RECUR_DATE_ RANGE	This event recurrence is invalid because its end time and duration do not match. {1}
INVALID_RECUR_DATE_ RANGE	This event recurrence is invalid because its end time is more than one day after its start time. {1}

Error Code Returned	Long Description or Message
INVALID_RECUR_DATE_RANGE	This event recurrence is invalid because its end time is not after its start time. {1}
INVALID_RECUR_DATE_RANGE	This event recurrence is invalid because its start time or end time/ duration is empty. {1}
INVALID_RECUR_DATE_RANGE	This event recurrence is invalid because its times are not in order. {1}
INVALID_RECUR_DESC_REQD	This event recurrence is invalid because it has no description. {1}
INVALID_RECUR_DOW	This event recurrence has an invalid day-of-week field. {1}
INVALID_RECUR_DOWIM	This event recurrence has an invalid day-of-week-in-month value. {1}
INVALID_RECUR_DOWMASK	This event recurrence is invalid because its day-of-week mask is not 7 characters long. {1}
INVALID_RECUR_FREQUENCY	This event recurrence has an invalid frequency. {1}
INVALID_RECUR_PATTERN	This event does not have a valid recurrence pattern.
INVALID_RECUR_PATTERN	This event recurrence is invalid because it is not a monthly or yearly event and it has day-of-week and day-of-week-in-month field values. {1}
INVALID_RECUR_PATTERN	This event recurrence is invalid because it only has one recurrence time and it must have either none or at least two. {1}
INVALID_RECUR_PATTERN	This event recurrence is invalid because one of its times is out of the range 0 to 86399. {1}
INVALID_RECUR_PATTERN	This event recurrence is invalid because only one of the day-of-week and day-of-week-in-month fields is set. Both must be set or both must be unset. {1}
INVALID_RECUR_PATTERN	This event recurrence is invalid either because it is not weekly and it has a day-of-week mask, or it is weekly and it has no day-of-week mask. {1}
INVALID_RECUR_PATTERN	This single day event is invalid since it contains a recurrence pattern.
INVALID_RECUR_PERIOD	This event recurrence has an invalid period. {1}
INVALID_RECUR_TIME_ZONE_REQD	This event recurrence in invalid because it has no time zone. {1}
INVALID_REF_CANT_INITIALIZE	Cannot initialize customerRefund: invalid creditMemo reference {1}.
INVALID_REF_CANT_INITIALIZE	Cannot initialize customerRefund: invalid customer reference {1}.
INVALID_REF_CANT_INITIALIZE	You can not initialize {1} by referencing {2}.
INVALID_REF_KEY	Invalid externalId {1}.
INVALID_REF_KEY	Invalid reference key [{1}].
INVALID_REFFERER_EMAIL	The refferer email address you have entered is not valid. Please try again.
INVALID_REFUND_AMT	Refund amount must be between zero and the original amount. Correct amount and retry request.

Error Code Returned	Long Description or Message
INVALID_REFUND_AMT	The amount you can refund is {1} because the the order has already been refunded for {2}
INVALID_REPORT	The referenced Report and Row are no longer valid, because the layout containing them has changed. Please edit this reference row to reselect Report and Row.
INVALID_REPORT_ID	Invalid report ID.
INVALID_REPORT_ROW	Invalid Reference Row
INVALID_REPORT_ROW	Invalid Report Row Reference
INVALID_REQUEST	invalid request (failed isValid() check). Email request handler unable to service request for address= <{1}> .
INVALID_RESOURCE_TIME	Total resource time for '{1}' cannot exceed {2} planned time entries.
INVALID_RESULT_ SUMMARY_FUNC	The result field {1} cannot be grouped. Please edit the search and omit this field or use a different summary function.
INVALID_RETURN_DATA_ OBJECT	The return value of the RESTlet function cannot be used to create a well-formed XML document.
INVALID_RETURN_DATA_ OBJECT	The XML document does not contain a root element because the RESTlet function does not return a JavaScript Object.
INVALID_REV_REC_DATE_ RANGE	Rev rec end date can not be before rev rec start date.
INVALID_ROLE	The specified role is invalid.
INVALID_ROLE	Your role does not give you permission to view this page.
INVALID_ROLE_FOR_EVENT	You seem to have been invited to this {1} in a different role. Please change your role to view the {1}.
INVALID_RQST_CONTACTS_ EXIST	it has associated primary contacts.
INVALID_RQST_PARENT_ REQD	it has associated contact records that would be left with no parent company.
INVALID_RQST_SBCUST_ JOBS_EXIST	it has associated sub-customers or jobs.
INVALID_SAVED_SRCH	Missing or invalid saved search for Custom KPI. The search must have a date column as an available filter. Please see help next to Custom KPI dropdown list on KPI setup page.
INVALID_SAVEDSEARCH	A saved search with the internal ID {1} does not exist.
INVALID_SAVEDSEARCH	We cannot return search columns for summary saved search {1}.
INVALID_SCHDUL_AMT	The total amount on the schedule must equal the sum of the individual recognition amounts.
INVALID_SCHDUL_AMT	The total amount on the schedule must be equal to the amount of the source transaction line.
INVALID_SCHDUL_FORMAT	To create a valid schedule, please enter the bracket values in ascending orders without gaps.
INVALID_SCRIPT_ID	A saved search with the script ID {1} does not exist.
INVALID_SEARCH	That search or mass update does not exist.

Error Code Returned	Long Description or Message
INVALID_SEARCH	You may search by {1} or {2} but not both
INVALID_SEARCH_CRITERIA	Can't search transactions: invalid cross reference key
INVALID_SEARCH_CRITERIA	Global Search supports at most three keywords and requires at least one. Keywords are composed of only letters, digits, and dashes.
INVALID_SEARCH_FIELD_KEY	search field keys are not consistent({1}/{2})
INVALID_SEARCH_FIELD_NAME	search field names are not consistent({1}/{2})
INVALID_SEARCH_FIELD_OBJ	{1} is not a valid search custom field
INVALID_SEARCH_FIELD_OBJ	{1} must be used to search custom field {2}
INVALID_SEARCH_FIELD_OBJ	Server application error: invalid search customfield object.
INVALID_SEARCH_FIELD_OBJ	Invalid search field object: {1}
INVALID_SEARCH_JOIN_ID	Invalid Search Join ID
INVALID_SEARCH_MORE	Invalid searchMore operation. Please make sure that you have had a successful search operation before you can perform any searchMore operation.
INVALID_SEARCH_OPERATOR	You need to provide a valid search field operator.
INVALID_SEARCH_OPERATOR	You can not use this operator '{1}' for internalId search.
INVALID_SEARCH_PAGE_INDEX	Invalid search page index.
INVALID_SEARCH_PAGE_SIZE	Invalid search page size.
INVALID_SEARCH_PREF	You cannot set returnSearchColumns to false while you specify search columns.
INVALID_SEARCH_PREF	You cannot set returnSearchColumns to true without specifying search columns or referencing a saved search.
INVALID_SEARCH_SELECT_OBJ	Invalid search select field object: {1}
INVALID_SEARCH_VALUE	You need to provide a search value.
INVALID_SEARCH_VALUE	You need to provide search values.
INVALID_SECONDARY_EMAIL	Invalid secondary email address. The email address must be in a valid format.
INVALID_SECPAY_CREDENTIALS	The username or password used to process the transaction with SECPay was not valid. Please make sure you have entered the correct username, password, and remote password in your SECPay account setup.
INVALID_SERIAL_NUM	No items match the entered serial number
INVALID_SERIAL_OR_LOT_NUMBER	Serial and lot numbers may not contain the '{1}' character.
INVALID_SESSION	A valid NLSession is required to generate record xml
INVALID_SESSION	A valid session is required. Please log in first.

Error Code Returned	Long Description or Message
INVALID_SHIP_DATE	The Future Ship Date is invalid. Please verify the entered Future Ship Date is no more than 7 days in the future, and resubmit the fulfillment.
INVALID_SHIP_FROM_STATE	The Ship From State/Province Code is missing or invalid. Please enter the 2 to 5 character abbreviation for the state or province of the address that contains it.
INVALID_SHIP_GRP	You cannot add shipping groups when creating a transaction that has multiple shipping routes enabled. You must first add the items, then get the transaction and update the shipping groups separately.
INVALID_SHIP_SRVC	The selected service is not valid for international shipments. Please choose an international service and retry your request.
INVALID_SHIP_TO_SATE	The Ship To State/Province Code is missing or invalid. Please enter the 2 to 5 character abbreviation for the state or province of the address that contains it.
INVALID_SHIPPER_STATE	The Shipper State/Province Code is missing or invalid. Please enter the 2 to 5 character abbreviation for the state or province of the address that contains it.
INVALID_SITE_CSTM_FILE	File is not a NetSuite site customization export file: it cannot be imported.
INVALID_SOAP_HEADER	Invalid SOAP Header: '{1}'. Value is '{2}'.
INVALID_SRCH	That search or mass update does not exist (internal id={1})
INVALID_SRCH_CRITERIA	The field rule value "{1}" is invalid for field type {2} with criterion "{3}."
INVALID_SRCH_CSTM_FLD	This search refers to custom field with id = {1} which either is restricted or is not applied to this record type.
INVALID_SRCH_FUNCTN	An nlobjSearchColumn contains an invalid function: {1}.
INVALID_SRCH_SORT	An nlobjSearchColumn that is not sortable contains a sort specification: {1}.
INVALID_SRCH_SUMMARY_ TYP	An nlobjSearchFilter contains an invalid summary type: {1}.
INVALID_SRCH_TYP	Search Type not allowed as standalone search
INVALID_SRVC_ITEM_SUB	The service item '{1:service item}' is not available in the subsidiary of customer '{2:customer}'.
INVALID_SSO	Invalid SuiteSignOn reference: {1}. That SuiteSignOn object does not exist or has been marked as inactive.
INVALID_SSS_DEBUG_ SESSION	You have exceeded the maximum allowable idle time for debugging scripts. To debug another script, simply reload the script debugger page and start a new debugging session.
INVALID_SSS_DEBUG_ SESSION	You have canceled your current script debugging session.
INVALID_STATE	Merchant State Length Error. Provide a valid state.
INVALID_STATE	Signup prospect state '{1}' is invalid.
INVALID_STATUS	You may not change this issue''s status from ''{1}'' to ''{2}''.
INVALID_SUB	The subsidiary restrictions on this record are incompatible with those defined for account: {1}. Subsidiary access on this record must be a subset of those permitted by the account.

Error Code Returned	Long Description or Message
INVALID_SUB	The subsidiary restrictions on this record are incompatible with those defined for account: {1}. Subsidiary access on this record must be a superset of those permitted by the account.
INVALID_SUB	The subsidiary restrictions on this record are incompatible with those defined for department: {1}. Subsidiary access on this record must be a subset of those permitted by the department.
INVALID_SUB	The subsidiary restrictions on this record are incompatible with those defined for item: {1}. Subsidiary access on this record must be a superset of those permitted by the item.
INVALID_SUB	The subsidiary restrictions on this record are incompatible with those defined for location: {1}. Subsidiary access on this record must be a subset of those permitted by the location.
INVALID_SUB	The Subsidiary selected doesnt match the bank account selected.
INVALID_SUB	This record does not support multiple subsidary restrictions. You must choose a single subsidiary.
INVALID_SUB	Transaction references multiple subsidiaries
INVALID_SUB	You may not add members to a group/kit/assembly unless the subsidiaries for those members completely contain the subsidiaries of the group/kit/assembly.
INVALID_SUB	{1} can not be used with the selected subsidiary
INVALID_SUBLIST_DESC	Invalid sublist description - all sublists must appear exactly as they do in the WSDL (eg end with 'List')
INVALID_SUBSCRIPTION_ STATUS	You cannot change the global subscription status from its current value of {1:status name}.
INVALID_SUBSCRIPTION_ STATUS	You cannot set the global subscription status to the value {1:status name}.
INVALID_SUMMARY_SRCH	In a summary search, you must sort by a result field with a summary function. Please go back and correct the sort by field on the results tab.
INVALID_SUPERVISOR	Employees can not be their own supervisor.
INVALID_SUPERVISOR	You can't insert this employee record as it would create a loop in the supervisor hierarchy.
INVALID_TASK_ID	The task ID: {1} is not valid. Please refer to the documentation for a list of supported task IDs.
INVALID_TASK_ID	You have specified an invalid task Id
INVALID_TAX_AMT	Invalid tax amount. Correct tax amount and retry request.
INVALID_TAX_CODE	Invalid Canadian Tax Code: {1}
INVALID_TAX_CODE_FOR_ SUB	The selected tax code is not available in subsidiary.
INVALID_TAX_CODES	Invalid Tax Code(s): {1}
INVALID_TAX_PMT	You may not commit tax payment information prior to the start date of the Payroll Service.
INVALID_TAX_VALUE	GST and PST amount cannot be negative!
INVALID_TAX_VALUE	GST tax value is not a valid number: {1}

Error Code Returned	Long Description or Message
INVALID_TIME_FORMAT	{1} is not a valid time and it should use the following format h:mm a.
INVALID_TO_DATE	invalid 'to' date
INVALID_TRACKING_NUM	The tracking number is not valid.
INVALID_TRACKING_NUM	You have entered a tracking number that exceeds the maximum size of {1} characters: {2}. Multiple tracking numbers must be separated by spaces, tabs, or commas. Slash (/), semicolon (;), colon (:), or any other character that is not a space or a comma will be interpreted as a part of the tracking number.
INVALID_TRAN_ITEM_LINE	Item {1} can not be included on the {2} because it is not distributed as of the transaction date
INVALID_TRANS	This transaction is not valid.
INVALID_TRANS_COMPNT	You have entered an invalid component for this transaction.
INVALID_TRANS_ID	Invalid Transaction ID. Correct the transaction ID, then re-submit.
INVALID_TRANS_ID	Only sale transactions can be refunded. Provide a valid transaction ID.
INVALID_TRANS_SUB_ACCT	Transaction subsidiary {1} is not valid for account {2}. Please choose a different account.
INVALID_TRANS_SUB_CLASS	Transaction subsidiary {1} is not valid for class {2}. Please choose a different class.
INVALID_TRANS_SUB_DEPT	Transaction subsidiary {1} is not valid for department {2}. Please choose a different department.
INVALID_TRANS_SUB_ENTITY	Transaction subsidiary {1} is not valid for entity {2}. Please choose a different entity.
INVALID_TRANS_SUB_ITEM	Transaction subsidiary {1} is not valid for item {2}. Please choose a different item.
INVALID_TRANS_SUB_LOC	Transaction subsidiary {1} is not valid for location {2}. Please choose a different location.
INVALID_TRANS_TYP	Transaction type specified is incorrect.
INVALID_TRANSACTIO_DATE	There are no Accounting Periods that cover this transaction date.
INVALID_TRANSACTION_DATE	Transaction date {1} is not valid. Transaction dates may be at most {2} years in the past and {3} years in the future.
INVALID_TRIAL_TYP	The trialtype is not availabe in the product specified.
INVALID_TYP	Invalid type {1}, use {2}
INVALID_UNIT_TYP	On serialized items, you may not choose a units type that has fractional conversion rates.
INVALID_UNSUPRTD_RCRD_TYP	Invalid or unsupported record type: {1}
INVALID_UPS_ACCT	An invalid UPS Account Number was entered. Please verify you have entered the correct Shipper Number and re-submit the form.
INVALID_UPS_PACKG_WEIGHT	UPS requires a minimum package weight of .1 LBS and a maximum package weight of 150 LBS. Please adjust the package weights accordingly and resubmit the fulfillment.

Error Code Returned	Long Description or Message
INVALID_UPS_VALUES	UPS did not accept the entered values for the following fields. Please go back and correct these values:
INVALID_URL	Please begin the {1} url with <b>http://</b> or <b>https://</b><p>Examples of a valid {1}url are: <b>http://www.mydomain.com/image.gif</b> or <b>https://one.two.org/user-name/test.jpg</b>
INVALID_URL	Request for invalid URL: {1}
INVALID_URL_PARAM	Error: URL param {1}="{2}" - expected an integer.
INVALID_VALUE	You have entered an invalid value {1} for {2}.
INVALID_VAT_AMOUNT	VAT amount cannot be negative
INVALID_VAT_REGSTRTN_NUM	Invalid VAT registration number {1}.
INVALID_VSOE_ALLOCTN	VSOE allocations must be greater than or equal to 0
INVALID_WEBSITE_SECTION	The Web site section you entered does not exist.
INVALID_WO	You have an invalid work order {1} or the order is already closed.
INVALID_WO_ITEM	Special Work Order Items can not be Drop Ship or Special Order
INVALID_WORLDPAY_ID	Exchange source does not recognize your WorldPay ID. Please check that it is correct.
INVALID_WS_VERSION	The async operation has been submitted through different endpoint version: {1}.
INVALID_YEAR	Invalid year {1}
INVALID_YEAR_FORMAT	Illegal year format or value. Examples: 1999, 2000, 2001, etc.
INVALID_ZIP_CODE	Merchant Zip Length Error. Provide a valid 5 digit zip.
INVALID_ZIP_FILE	Invalid archive. Zip file must contain at least one file.
INVALID_ZIP_POST_CODE	The submitted Zip/Postal Code is invalid. This field may only contain a maximum of 16 digits, spaces, and the dash character (-).
INVENTORY_NUM_DISALLWD	Inventory numbers are only allowed on items with serial numbered or lot numbered items.
INVLAID_BOOLEAN_VALUE	You have entered an invalid boolean value. Please use true, false, T, or F for boolean values and resubmit your import.
IP_REQUEST	Your IP address {1} does not match any of the ipaddress rules specified for this account.
ISSUE_ASSIGNEE_DISALLWD	The specified assignee is disallowed for this issue's status.
ISSUE_PRODUCT_VERSION_MISMATCH	Cannot set issue {1} to {2} {3} and {4} {5} because that product is not associated with that version.
ISSUE_VERSION_BUILD_MISMATCH	Issue version and build do not match.
ITEM_ACCT_REQD	One of the items on this transaction has an amount but no account. Please fix the item and resubmit the transaction.
ITEM_ACCT_REQD	One of the items on this transaction has an amount but no account. Please fix the item and resubmit the transaction. It might be that you have recently elected to charge for shipping and have not assigned an account to the shipping item that is included in this transaction.

Error Code Returned	Long Description or Message
ITEM_ACCT_REQD	You must specify asset and COGS accounts for this inventory item.
ITEM_COUNT_MISMATCH	COGS_CORRECTION: 2 means of calculating the item count do not match for item: {1} vs {2})
ITEM_COUNT_MISMATCH	COGS_CORRECTION: 2 means of calculating the item count do not match for item: {1} vs {2}) There are transactions in the system in which this item is used but the asset account for that item is not the current Asset Account in the item record
ITEM_IS_UNAVAILABLE	(Item is unavailable)
ITEM_NAME_MUST_BE_ UNIQUE	An item with that name already exists. Please choose another name
ITEM_NOT_UNIQUE	The item [{1}] is not unique.
ITEM_PARAM_REQD_IN_URL	Error - Item parameter (id=nnn) was not provided on the URL
ITEM_QTY_AMT_MISMATCH	Inventory and assembly items cannot have zero quantity and non-zero amount.
ITEM_TYP_REQS_UNIT	Items of type {1} require {2} unit
ITEM_TYP_REQS_UNIT	Items of type {1} subtype {2} require {3} unit
JE_AMOUNTS_MUST_ BALANCE	The amounts in the journal entry must balance.
JE_LINE_MISSING_REQD_ DATA	{1} are mandatory on all lines of the journal entry.
JE_MAX_ONE_LINE	Journal Entries can have a maximum of {1} lines.
JE_REV_REC_IN_PROGRESS	This account is currently processing Revenue Recognition Journal Entries. Only one such process is allowed at a time.
JE_UNEXPECTED_ERROR	Journal Entries failed to be created due to unexpected error.
JOB_NOT_COMPLETE	The specified job is not complete yet
JS_EXCEPTION	A JavaScript Exception was thrown
KEY_REQD	Empty key not allowed for {1}
KPI_SETUP_REQD	Please enable the Forecast ({1}) & Quota KPIs
KPI_SETUP_REQD	Please enable the Forecast & Quota KPIs
KPI_SETUP_REQD	please enable the Sales & Forecast KPIs
KPI_SETUP_REQD	Please enable the Sales & Forecast ({1}) KPIs
KPI_SETUP_REQD	Please enable the Sales & Quota KPIs
LABEL_REQD	Please enter a value for Label
LANGUAGE_SETUP_REQD	Please go to company preference to add language to translate.
LINK_LINES_TO_ONE_ORD	Lines on this record can only be linked to a single order.
LINKED_ACCT_DONT_ MATCH	You are attempting to link transaction line items, but items on the lines do not match. This can happen when you create a fulfillment from a sales order, a receipt from a purchase order, an invoice from a sales order, a vendor bill from a purchase order, or a reimbursement from a purchase. Please verify that items in the transaction you are creating match the items in the originating transaction.

Error Code Returned	Long Description or Message
LINKED_ENTITIES_DONT_ MATCH	You are attempting to link transactions, but the entities on the transactions do not match.
LINKED_ITEMS_DONT_ MATCH	Linked items don't match
LIST_ID_REQD	Required field missing in a related list. You must set {1}.
LIST_KEY_REQD	There is no list key for field {1} of list {2}. Please assign a key and resubmit your task.
LOCATIONS_IN_USE	Your classes cannot be converted to locations because your existing location records are referred to by transactions or other records. These location records cannot be overwritten.
LOCATIONS_SETUP_REQD	You must first define locations (Lists->Locations->New) before you can distribute inventory.
LOCATIONS_SETUP_REQD	You must first define locations (Lists->Locations->New) before you can transfer inventory.
LOCATION_REQD	You must specify a location to use {1} numbers when Multi-Location Inventory is enabled
LOCKED_DASHBOARD	Your dashboard has been set up and locked by an administrator. Please contact them for details.
LOGIN_DISABLED	Invalid login. Customer access is disabled.
LOGIN_DISABLED	Login access has been disabled for this role.
LOGIN_DISABLED	Your access to {1} has been deactivated. Please contact the company's administrator to re-activate your access.
LOGIN_DISABLED	Your access to this account has been removed or disabled. Please contact the account adminstrator.
LOGIN_DISABLED_PARTNER_ CTR	Disabled login: Advanced Partner Center access has been disabled by the account administrator.
LOGIN_DISABLED_PARTNER_ CTR	Disabled login: Standard Partner Center access has been disabled by the account administrator.
LOGIN_EMAIL_REQD	Invalid login. You must provide an email address.
LOGIN_NAME_AND_PSWD_ REQD	Please enter both a user name and a password.
LOGIN_REQD	You must <a href='/pages/login.jsp' target='_self'>log in</a> before accessing this page.
LOST_UPSELL_CRITERIA	Your upsell criteria were lost. This is probably due to a transient condition such as a server reboot. Click <a href=# onclick='history.go(-1);'>here</a> to go back and try again.
MACHN_LIST_KEY_NAMES_ REQD	Server application error: no list key names are defined for field {1} of record of type {2}.
MANDATORY_PRD_TYPE_ REQD	Please select the mandatory period type...
MASS_UPDATE_CRITERIA_ LOST	Your mass update criteria were lost. This is probably due to a transient condition such as a server reboot. Click <a href=# onclick='history.go(-1);'>here</a> to go back and try again.

Error Code Returned	Long Description or Message
MATCHING_CUR_SUB_REQD	The parent specified must have the same currency and subsidiary as the child
MATCHING_SERIAL_NUM_REQD	The serial numbers on a transfer order receipt must have been fulfilled
MATRIX_INFO_TEMP_LOST	Matrix item information was lost. This was probably due to a transient condition like a server reboot. Please try again.
MATRIX_SUBITEM_NAME_TOO_LONG	The following matrix sub-item name is too long (80 character max):<p> {1} <p> Please shorten your parent item name or your option abbreviations.
MAX_16_LINES_ALLWD_PER_BILLPAY	A maximum of 16 lines per payee can be applied per online bill payment.
MAX_200_LINES_ALLWD_ON_TRANS	Journal Entries can have a maximum of 200 lines.
MAX_BARCODE_PRINT_EXCEEDED	A maximum of {1} barcodes can be printed at a time.
MAX_BULK_MERGE_RCRDS_EXCEEDED	You cannot perform a bulk merge operation with a group larger than {1} records
MAX_EMAILS_EXCEEDED	This campaign email event exceeds the number of emails ({1}) that can be sent per month without setting up a default campaign domain or specifying one on the campaign email template.
MAX_EMAILS_EXCEEDED	This merge operation exceeds the number of emails ({1}) that can be sent per month without setting up a bulk merge domain or specifying one on the email template.
MAX_RCRDS_EXCEEDED	The maximum number ({1}) of records allowed for a {2} operation has been exceeded.
MAX_VALUES_EXCEEDED	Too many values specified for a MultiSelectField, the maximum is 1000
MEDIA_FILE_INVALID_JSCRIPT	Media file was of type javascript and would not compile. Error on line:
MEDIA_NOT_FOUND	Media item not found {1}
MEDIA_NOT_INITIALIZED	Media Item cannot be initialized
MEMORIZED_TRANS_ERROR	failed retrieving 'chargeit' record from trancard while processing memorized tran
MEMORIZED_TRANS_ERROR	A failure occurred while trying to enter one of your automated memorized transactions. The reason for this failure is shown in the description field on this page. Because of the failure, the system has changed this memorized transaction from an automated one to a reminder. You may click the link above to view the memorized transaction in question and make any needed changes to it. When you are done with the changes, click Memorize and then Cancel. This will modify your existing memorized transaction rather than creating a new one. When filling out the Memorized Transaction Form, you may choose to select 'Automatic' to resume automated posting of this transaction.
MERGE_OPERATION_DISALLWD	You cannot perform merge operations on records that belong to your group.
MERGE_RCRD_REQD	You must specify a record to merge into

Error Code Returned	Long Description or Message
METAVANTE_ERROR	This Online Bill Payment has been {1} in your NetSuite account. However, the payment was not able to be {2} at Metavante. <BR>To {3} this payment at Metavante, please contact NetSuite Customer Support immediately, using the options provided in the Support > Customer Service section of your NetSuite account.
METAVANTE_SECRET_ANSWER_REQD	Missing Secret Answer. A secret answer is required by the Metavante CSP service. It cannot be null or empty.
METAVANTE_SECRET_QESTION_REQD	Missing or invalid Secret Question ID. A valid Secret Question is required by the Metavante CSP service. Please refer to Table 2, above, for a list of valid Secret Question IDs.
METAVANTE_SETUP_REQD	Your NetSuite account is not currently integrated with a Metavante Online Bill Pay account. To set up an active account, you need to reapply to Metavante. Go to Setup > Set Up Online Bill Pay and follow the instructions on that page to apply for a Metavante account.
METAVANTE_TEMP_UNAVAILBL	Metavante is temporarily unavailable. Please try again later.<BR><BR><BR>If you would like to print the payment to mail yourself, click Back, and then click the date of the payment on the Approve Online Bill Payments page. When the payment's detail record appears, clear the Bill Pay box and either check the To Be Printed box and click Submit or click the Print button.
MISMATCH_EVENT_ISSUE_STATUS	Event status ({1}) and issue base status ({2}) do not match
MISMATCH_ISSUE_PRODUCT_VERSION	Issue product and version do not match.
MISMATCH_SALES_CONTRIBUTION	Sales team sales rep total does not equal 100%, {1} sales reps, {2} total contribution.
MISMATCHED_CURRENCY	The transaction currency does not match the names currency
MISMATCHED_QTY_PRICING	Quantities do not match accross pricings
MISMATCHED_SEARCH_PARENTHESIS	Search error: Parentheses are unbalanced.
MISSING_ACCT_PRD	You are attempting to create an amortization or revenue recognition schedule outside the range of available accounting periods. Please adjust the periods on this transaction or go to Setup>Accounting>Manage Accounting Periods to set up more periods.
MISSING_CRNCY_EXCH_RATE	No currency conversion rate defined
MISSING_ENUM	No Enumerated Value {1} for Enumerated Type {2}
MISSING_REQD_FLD	Missing required value for mandatory field: {1}
MISSING_REQD_FLD	The Company record does not have all required fields set. Please ensure the State, Zip/Postal Code, and Country fields are set and try your request again.
MISSNG_ACCT_PRD	Unable to find an Accounting Period for the allocation date.
MISSNG_REV_REC_RCRD	Unable to locate Revenue Recognition records.
MISSNG_SO_REV_REC_PARAMS	Unable to get Revenue Recognition parameters from originating sales order.

Error Code Returned	Long Description or Message
MISSNG_SO_START_END_DATES	Unable to acquire start and end date from Sales Order.
MLI_REQD	Multi-location Inventory Error (MLI_LOCATION_REQUIRED): this transaction or its items must have locations.
MLTPLE_TAX_LINES_DISALLWD	Multiple Tax lines for line item in transaction:
MSNG_FIELD_OWRTE_MUST_BE_TRUE	The missingFieldOverwrite attribute must be true when updating a salesOrder.
MST_UPDATE_ITEMS_THEN_RATES	You cannot update items and shipping rates at the same time on transactions that have multiple shipping routes enabled. You must first update the items, then get the transaction and update the shipping rates separately.
MULTI_ACCT_CANT_CHANGE_PSWD	The password cannot be changed here because the email address is associated with multiple accounts. The user must change their password via the link in the settings portal of the home page.
MULTI_LOC_INVT_ERROR	Multi-Location Inventory Error: You may not create an Assembly Build transaction with an assembly item that has not been distributed and member items that have been distributed. You must create an Inventory Distribution transaction for the assembly item before building it. You also may not create an Assembly Build transaction on a date prior to the distribution date of the assembly but after the distribution date of any of the member items.
MULTI_PRIMARY_PARTNER_DISALLWD	You are not allowed to select multiple primary partners.
MULTI_SHIP_ROUTES_REQD	{1} {2} has multiple shipping routes enabled, which is only supported in version 2008_2 and newer. You are not allowed to update any shipping fields on this record.
MULTISELECT_TYPE_REQD	Server application error: no multiselect type is defined for field {1} of {2} record type.
MUST_DEFINE_BASE_UNIT	One unit must be designated as the base unit.
MUST_RESUBMIT_RCRD	Configuration changes have been made to your account. You must resubmit your record.
NAME_ALREADY_IN_USE	A mass update has already been saved with that name. Please use a different name.
NAME_ALREADY_IN_USE	A search has already been saved with that name. Please use a different name.
NAME_REQD	Missing Name. Name is a required field and it cannot be null or empty.
NAME_TYPE_FLDR_FIELDS_REQD	missing required fields : name, type, and folder
NARROW_KEYWORD_SEARCH	Please provide more detailed keywords so your search does not return too many results.
NEED_BILL_VARIANCE_ACCT	Bill variance account is missing
NEGATIVE_PAYMENT_DISALLWD	Negative payments not allowed

Error Code Returned	Long Description or Message
NEGATIVE_TAX_RATE_ DISALLWD	A Tax rate cannot be negative
NEW_CONNECTION_ DISALLWD	Not allowed to create new connections.
NEXUS_REQD	No tax agency defined for subsidiary: subsidiary {1} is not linked to nexus {2}
NO_DATA_FOUND	No data was found
NO_EXPENSES_FOR_PRD	The Allocation sources or destinations did not have any expenses associated with them for the selected period.
NO_ITEMS_TO_PRINT	There are no items to print
NO_MASS_UPDATES_ RUNNING	There are currently no mass updates running.
NO_MTRX_ITEMS_TO_ UPDATE	There are no matrix subitems to update.
NO_ORD_SHPMNT	There is no shipment on that order.
NO_RCRD_FOR_USER	There is no record for this user in the company's entity table. (emaillogin.semail='{1}', kentity={2})
NO_RCRDS_MATCH	No Records matched your request.
NO_SCHDUL_APPLIED	There were no schedules that need to applied to the particular period.
NO_SCHDUL_APPLIED	There were no schedules that need to be applied to the input accounting period.
NON_ADMIN_CANNOT_ INITIATE_LINK	This user cannot integrate with a partner.
NONMATCHING_EMAILS	Email addresses don't match
NONUNIQUE_INDEX_VALUE	An attempt was made to insert a row with a non-unique index value. Dynamic SQL being executed [{1}]
NONZERO_AMT_REQD	You did not enter non-zero amounts for any accounts.
NONZERO_QTY_REQD	Build quantity must be greater than zero.
NONZERO_WEIGHT_REQD	Selected service must have a weight greater than zero.
NOT_AN_INVITEE	You are not on the invitee list for event.
NOT_IN_INVT	You may not distribute {1} numbers that are not currently in inventory. You attempted to distribute the following {1} numbers that were not in inventory: {2}
NULL_CHECK_NUMBER	Null Check Number
NUM_ITEMS_GRTR_THAN_ QTY	The number of {1} entered ({2}) is greater than the item quantity ({3})
NUM_ITEMS_NOT_EQUAL_ TO_QTY	The number of {1} entered ({2}) is not equal to the item quantity ({3})
NUM_REQD_FOR_FIRST_ LABEL	No number was specified for the first label.
NUMERIC_CHECK_NUM_ REQD	Invalid Check Number. Check number must be a numeric value and can be at most 7 digits long.

Error Code Returned	Long Description or Message
NUMERIC_REF_NUM_REQD	Reference Number Must Be Numeric. Please provide a valid number and re-submit.
OI_FEATURE_REQD	You have not enabled Outlook Integration feature for your account.
OI_PERMISSION_REQD	You do not have permission to access Outlook Integration feature.
ONE_ADMIN_REQD_PER_ACCT	This operation would leave your account without an active Administrator. To successfully perform the mass update, please deselect at least one entity with an Administrator role.
ONE_ADMIN_REQD_PER_ACCT	You can't delete this employee. No administrators for this account would remain.
ONE_ADMIN_REQD_PER_ACCT	You can't inactivate {1}. The account would be left with no active administrators.
ONE_ADMIN_REQD_PER_ACCT	You can't remove the administrator role from this user. No administrators for this account would remain.
ONE_EMPL_REQD	At least one employee is required to process payroll
ONE_PAY_ITEM_PER_EMPL	An employee may not have multiple instances of payroll items. Go <a href="javascript:history.go(-1);";>back</a>, remove these items and resubmit.
ONE_POSITIVE_VALUE_REQD	You must enter at least one positive value for at least one item.
ONE_RCRD_REQD_FOR_MASS_UPDATE	Please create at least one {1} before using this mass update.
ONE_ROLE_REQD	You can't inactivate all roles. You would not be able to log in.
ONLINE_BANK_FILE_REQD	You must first upload an Online Bank file before using the Online Bank Statement.
ONLINE_BILL_PAY_DUP	Another entity with the same online bill pay name, account, and address already exists. Please find and use that entity for online bill pay.
ONLINE_BILL_PAY_SETUP_REQD	<b>{1}</b> is not set up for Online Bill Pay. To set up this payee, click Go Back. When the Approve Online Bill Payments page appears, click Enable Payee in the Enabled for Billpay column. When the payee's record appears, check Enable Online Bill Pay and submit these required fields:<ul><li>Legal Name</li><li>Print on Check As</li><li>Phone</li><li>Billing Address</li></ul> Then, go to Transactions > Approve Online Bill Payments to approve the payment.
ONLINE_FORM_DSNT_EXIST	This online form does not exist.
ONLINE_FORM_EMPTY	The online form you requested is empty.
ONLINE_FORM_ID_REQD	Missing required online form ID
ONLINE_FORM_USER_ACCESS_ONLY	This form is only accesible to online form users.
ONLINE_ORD_FEATURE_DISABLED	Can't open store for {1}. This company does not have the <b>Use Sales Orders</b> feature enabled. The feature is required for customers to make online purchases.
ONLY_ONE_CONTRIB_ITEM_REQD	Only one instance of a company contribution item is allowed on an employee record.
ONLY_ONE_DEDCT_ITEM_REQD	Only one instance of a deduction item is allowed on an employee record.

Error Code Returned	Long Description or Message
ONLY_ONE_DISTRIB_ALLWD	You may not distribute {1} numbers more than one time. You attempted to distribute the following {1} numbers more than one time: {2}
ONLY_ONE_EARNING_ITEM_REQD	Only one instance of an earning item is allowed on an employee record.
ONLY_ONE_LOT_NUM_ALLWD	You may not enter more than a single serial/lot number before an item is selected.
ONLY_ONE_PREF_BIN_ALLWD	There may be at most one preferred bin per location for an item. The following location has more than one preferred bin for this item: {1}
ONLY_ONE_PREF_BIN_ALLWD	You may not have more than one preferred bin per item.
ONLY_ONE_UNIT_AS_BASE_UNIT	Only one unit may be designated as the base unit.
ONLY_ONE_UPLOAD_ALLWD	You cannot upload more than one file at a time
ONLY_ONE_WITHLD_ITEM_REQD	Only one instance of a withholding item is allowed on an employee record.
OPENID_DOMAIN_IN_USE	This domain is already in use by another NetSuite account, and a domain cannot be used by multiple accounts.
OPENID_NOT_ENABLED	The OpenID Single Sign-on feature is not enabled in this account.
OPRTN_UNAVAILBL_TO_GATEWAY	
ORD_ALREADY_APPRVD	You cannot cancel this order because it has already been approved.
ORD_UNAVAILBL_TO_FULFILL	This order is not available for fulfillment. The current Google Order Financial State is {1}
ORDER_DSNT_EXIST	That order does not exist.
OTHER_PMT_AUTH_IN_PROGRESS	Another payment authorization is currently in progress for this order. Pleaase try again in a few minutes.
OVER_FULFILL_DISALLWD	You can not over-fulfill an item unless you have selected the 'Allow Overage on Item Fulfillments' preference.
OVER_FULFILL_RECEIV_DISALLWD	Transfer orders can not be overfulfilled or overreceived
OVERAGE_DISALLWD	Overage is not allowed.
OVERLAPPING_PRDS_DISALLWD	Illegal period structure. Overlapping periods.
OVERLAPPING_PRDS_DISALLWD	There is an overlapping period. Please check your Active or Inactive Periods to ensure that there is not an existing period.
OWNER_REQD	You cannot make a contact private without an owner
PACKAGE_WEIGHT_REQD	Attempted to create a package without specifying a nonzero package weight.
PACKG_LEVEL_REF_DISALLWD	Package level reference numbers are not allowed for shipments whose origin/destination pair is not US/US or Puerto Rico/Puerto Rico.
PACKG_VALUE_TOO_LARGE	Package declared value cannot be greater than $999.00 USD

Error Code Returned	Long Description or Message
PARENT_CANT_ITSELF_BE_MEMBER	Parent item can not be a member of itself
PARENT_MUST_BE_MATRIX_ITEM	A Child matrix item's parent must be a matrix item
PARENT_REQD	A Child matrix item must have its parent specified
PARSING_ERROR	Unable to parse value into the correct type. Problem occured on record type: {1}, internalid: {2}, field : {3}, value: {4}
PARTIAL_FULFILL_RCEIV_DISALLWD	Transfer orders can not be partially fulfilled or partially received
PARTNER_ACCESS_DENIED	Partners do not have access to this item.
PARTNER_ACCT_NOT_LINKED	Account ({1}) from partner ({2} is not linked.
PARTNER_CODE_ALREADY_USED	A partner with that partner code ({1}) already exists.
PAY_HOLD_ON_SO	This sales order cannot be fulfilled because it has a payment hold.
PAYCHECK_ALREADY_PAID	You are trying to edit a paycheck that is already paid by direct deposit. A paycheck cannot be edited after funds have been processed by the Automated Clearing House (ACH).
PAYCHECK_IN_USE	You cannot clear this paycheck because it is linked to by one or more liability payments. You must delete or void those transactions first
PAYEE_REQD_FOR_PMT	Your payment has been recorded, but an online bill pay payment will not be made because no payee was specified.You should return to the payment screen if you wish to print the check.
PAYPAL_FUND_SOURCE_REQD	Please return to PayPal to select a different funding source.
PAYPAL_INVALID_PMT_METHOD	Paypal is unable to process this payment. Please select an alternate payment method.
PAYPAL_INVALID_PMT_METHOD	Your PayPal account is not configured to use Express Checkout. Please follow directions on the PayPal payment method record.
PAYPAL_PMT_NOTIFICATION	PayPal Payment Notification
PAYPAL_SETUP_REQD	The account referenced by this paypal id is not setup to use express checkout. Please return to the paypal setup page and follow directions for setting up paypal express checkout.
PAYROLL_COMMITTED	You are trying to edit a paycheck reversal that is in a committed Payroll Batch.
PAYROLL_COMMITTED	You are trying to edit a paycheck that is in a committed Payroll Batch.
PAYROLL_COMMITTED	You are trying to modify a committed payroll batch or a document on a committed payroll batch.
PAYROLL_EXPENSE_ACCT_REQD	Please select an expense account for payroll item <a href='/app/common/item/payrollitem.nl?id={1}&e=T'>{2}</a>
PAYROLL_EXPENSE_ACCT_REQD	Please select a expense account for payroll item <a href='/app/common/item/payrollitem.nl?id={1}&e=T'>{2}</a>
PAYROLL_FEATURE_DISABLED	You have not enabled the Payroll feature.

Error Code Returned	Long Description or Message
PAYROLL_FEATURE_UNAVAILABLE	You are trying to edit a Pay Cheque - Payroll is not available in NetSuite Canada.
PAYROLL_IN_PROCESS	A payroll process is currently running in this account. Please try again in a few minutes.
PAYROLL_ITEM_DELETE_DISALLWD	Unable to remove payroll item: {1} - There are existing transactions for this payroll item. You may mark it inactive instead.
PAYROLL_LIABILITY_ACCT_REQD	Please select a liability account for payroll item <a href='/app/common/item/payrollitem.nl?id={1}&e=T'>{2}</a>
PAYROLL_MAINTENANCE	The Payroll feature in your account is undergoing routine maintenance from {1} Pacific Time on {2} to {3} Pacific Time on {4}. We apologize for any inconvenience this causes you.
PAYROLL_SETUP_REQD	The default payroll expense account is missing. Please go to Setup -> Payroll -> Set Up Payroll and set the default payroll expense account.
PAYROLL_SETUP_REQD	You are trying to create a paycheck before your payroll setup is complete. Please complete your payroll setup.
PAYROLL_SETUP_REQD	You are trying to process payroll before your payroll setup is complete. Please complete your payroll setup.
PAYROLL_UPDATE_REQD	You made changes that require you to update your payroll information. Click <a href="/app/payroll/managepayroll.nl">here</a> to commit updates to the payroll service.
PERMISSION_VIOLATION	Permission Violation: you may not access this record.
PERMISSION_VIOLATION	Permission Violation: you may no longer edit this record.
PHONE_NUM_REQD	Please provide a phone number.
PIN_DEBIT_TRANS_DISALLWD	Non-USD pin debit transactions are not supported. PINNED Debit for USD transactions only.
PLAN_IN_USE	This plan has already been used to generate commission calculations and can't be deleted.
PLAN_OVERLAP_DISALLWD	Plan overlap is not permitted. You have attempted to assign someone to this plan for a time period that overlaps with another plan.
PMT_ALREADY_APPRVD	The payment has already been approved and sent to the bill pay carrier for processing.
PMT_ALREADY_EXISTS	A payment with the same amount and date already exists for this payee.
PMT_ALREADY_SBMTD	This payment has already been submitted for online bill pay.
PMT_EDIT_DISALLWD	Access to this Bill Pay transaction is restricted, and it cannot be modified. Transactions can only be modified until 3PM CST on the payment date.
PMT_EDIT_DISALLWD	This liability payment cannot be edited while it has an Automated Clearing House transmission in process.</TD></TR><TR><TD class=text></TD></TR><TR><TD class=text> To view the status of payments with ACH transmissions, go to Transactions > View ACH Vendor Payments Status.
POSITIVE_BIN_QTY_REQD	Bin quantities must be positive or zero.
POSITIVE_QTY_REQD	Assembly member items must have positive quantities
POSITIVE_UNITCOST_REQD	Lines cannot have a negative unit cost.

Error Code Returned	Long Description or Message
POSTING_DISCOUNT_ DISALLWD	Posting Discounts are not allowed on lines with Revenue Recognition Schedules.
POSTING_PRD_SETUP_REQD	Creation of Journal Entries require a single Accounting Period value across all Revenue Recognition events. Please setup a 'Posting Period' filter.
PRDS_DISALLWD_NAMES_ NOT_UNIQUE	After adding new periods, not all names would be unique.
PRD_SETUP_REQD	You must change your period definitions to contain fiscal years. Please visit 'Setup->Manage Accounting Periods' and click 'Set Up Year'.
PRD_SETUP_REQD	You must define the periods of the prior fiscal year. Please visit 'Setup->Manage Accounting Periods' and click 'Set Up Year'.
PRDS_DISALLWD_NAMES_ NOT_UNIQUE	After adding new periods, not all names would be unique.
PREF_VENDOR_COST_REQD	Drop ship/Special Order items must have a preferred vendor and a purchase price.
PREF_VENDOR_REQD	Drop ship/Special Order items must have a preferred vendor for each of the {1} the item is accessible to.
PREFERRED_TAX_AGENCY_ REQD	A preferred Tax Agency has been deleted - Please choose a new one in <a href="/app/setup/acctsetup.nl";>Set Up Accounting</a>
PREFERRED_TAX_AGENCY_ REQD	Error: No preferred Tax Agencies have been set up (go to <a href='/app/setup/acctsetup.nl';>Set Up Accounting</a>)
PRIVATE_RCRD_ACCESS_ DISALLWD	You cannot view or edit this record because it is marked private
PRIVATE_STATUS_CHNG_ DISALLWD	You cannot make this contact private.
PSWD_EXPIRED	Password has expired. Please change your Netsuite password before continuing.
PSWD_REQD	A password must be entered when granting login access privileges to this record.
PSWD_REQD	Password is empty.
PSWD_REQD	Please type your password into both fields.
PSWD_REQD	You must provide a password to give this person access to your account.
PWSDS_DONT_MATCH	New passwords don't match.
PWSDS_DONT_MATCH	The passwords you entered do not match. Please reenter your passwords.
PWSDS_DONT_MATCH	The Passwords you entered do not match. Please reenter your passwords.
PWSDS_DONT_MATCH	The passwords you have entered do not match.
QTY_EXCEEDED_QTY_ BUCKETS	More quantities defined than there are quantity buckets
QTY_REQD	Quantities must be defined
RATE_REQUEST_SHPMNT_ REQD	The rate request shipment value has not been set.

Error Code Returned	Long Description or Message
RATE_SRVC_UNAVAILBL	The rate for this service is not available for the specified source and destination addresses.
RCRD_DELETED_SINCE_RETRIEVED	Items you have requested in the record have been deleted since you retrieved the form
RCRD_DELETED_SINCE_RETRIEVED	The record has been deleted since you retrieved it.
RCRD_DSNT_EXIST	Group Record Not Found
RCRD_DSNT_EXIST	That record does not exist.{1}
RCRD_DSNT_EXIST	There are no records of this type.
RCRD_EDITED_SINCE_RETRIEVED	The record has been edited since you retrieved it. Hit the back button and click Refresh/Reload to retrieve the updated record, then resubmit your changes.
RCRD_HAS_BEEN_CHANGED	Record has been changed
RCRD_ID_NOT_INT	Record id is not integer: {1}
RCRD_LOCKED_BY_WF	This record has been locked by a user defined workflow.
RCRD_NOT_FOUND	Could not find any records by this name.
RCRD_PREVSLY_DELETED	This record has already been deleted.
RCRD_PREVSLY_DELETED	This record has been deleted since the list was generated.
RCRD_REF_RCRD_TYP_MISMATCH	The record type and its object reference are not matched.
RCRD_SUB_MISMATCH_WITH_CLASS	The subsidiary restrictions on this record are incompatible with those defined for class: {1}. Subsidiary access on this record must be a subset of those permitted by the class.
RCRD_TYPE_REQD	The record type is required.
RCRD_UNEDITABLE	That record is not editable.
REACHED_LIST_END	You have reached the end of the list.
REACHED_LIST_START	You have reached the start of the list.
REC_TYP_REQD	You must provide either standard or custom record type information.
RECALCING_PLAN_SCHDUL	Cannot inactivate a plan when schedules in the plan are recalculating. Try again when recalculation is complete.
RECUR_EVENT_DISALLWD	A yearly event cannot be on the 29th of February
RECURSV_REF_DISALLWD	ERROR: Recursive Reference
REPORT_EXPIRED	Your report request has expired. A newly created report definition will normally expire after 15 minutes of inactivity unless the definition is saved. Click back to the report definition page and re-submit your report request.
REQD_FORM_TAG_MISSING	Your HTML template file is missing some mandatory fields or a form tag. Please make the changes to the form and try again
REQD_FORM_TAG_MISSING	Your Online HTML Form template is missing a required closing FORM tag
REQD_FORM_TAG_MISSING	Your Online HTML Form template is missing one or more of the required following required tags: <HTML>, <HEAD>

Error Code Returned	Long Description or Message
REQD_FORM_TAG_MISSING	Your Online HTML template is missing the required form tag for the main form: <NLFORM>
REQD_LOC_FIELDS_MISSING	Location {1} does not have all required fields set. Please ensure the State, Zip/Postal Code, and Country fields are set and try your request again.
REQD_SUB_FIELDS_MISSING	The Subsidiary {1} does not have all required fields set. Please ensure the State, Zip/Postal Code, and Country fields are set and try your request again.
REQUEST_PARAM_REQD	This request is missing a required parameter.
REV_REC_DATE_REQD	No Revenue Recognition Start Date Specified
REV_REC_TMPLT_DATA_ MISSING	One or more line items on this transaction have Variable Revenue Recognition Templates, but do not have the required {1} also populated. Please either change the Template for these items or indicate which {1} will be used to schedule the recognition of revenue.
REV_REC_UPDATE_ DISALLWD	Modification of revenue recognition related information on this item is not allowed because revenue has been recognized for this or related lines.
REVERSAL_DATE_WARNING	Reversal Date is in a closed accounting period. Please go to Manage Accounting Periods and re-open the accounting period.
ROLE_REQD	Please specify a role to which access should be granted
ROLE_REQUIRED	To login, a role is required unless a default has been previously set.
ROUNDING_DIFF_TOO_BIG	rounding difference too big -> tax1: {1} /tax2: {2}
ROUNDING_ERROR	Rounding Error: {1}
ROUTING_NUM_REQD	Missing Routing Number. Bank routing number is a required field and it cannot be null or empty.
SALES_DISCOUNT_ACCT_ REQD	Please set the Sales Discount Account Preference
SAME_ACCT_TYP_REQD_ FOR_PARENT	Parent acccount must be of same account type.
SAVED_SRCH_EMAIL_ERROR	E-mail Alert Failure using saved search '{1}'. Reason: {2}
SCHDUL_EDIT_DISALLWD	This schedule cannot be edited as it has already been used for commission calculations. Please go back and select 'save as new' instead.
SCHEDULED_REPORT_ ERROR	NetSuite encountered an error while trying to generate your scheduled report.
SCHEDULED_REPORT_ ERROR	NetSuite encountered an error while trying to generate your scheduled report. Please click {1}here{2} to notify NetSuite Support.
SCHEDULED_REPORT_ ERROR	NetSuite encountered an error while trying to generate your scheduled report. This error most likely occurred because the query was too large. To run this report successfully, try using filters, or select a different date/ period range to narrow the query.
SCHEDULED_REPORT_ ERROR	NetSuite encountered an error while trying to generate your scheduled report. This error most likely occurred because the query was too large. To run this report successfully, try using filters, or select a different date/ period range to narrow the query. If you continue to encounter errors, please click {1}here{2} to notify NetSuite Support.

Error Code Returned	Long Description or Message
SEARCH_DATE_FILTER_REQD	The search must have a date column as an available filter
SEARCH_ERROR	Unable to search - Unexpected error in search engine.
SEARCH_INTEGER_REQD	Please enter an integer number to search on.
SEARCH_TIMED_OUT	Your search has timed out. If your search includes the '{1}' operator, try using '{2}' instead. If your search includes broad search criteria, try narrowing the criteria.
SEARCH_TIMED_OUT	Your search has timed out. This might be avoided by using a smaller page size.
SECURE_TRANS_REQD_ON_CHECKOUT	Store Server Error: As configured, this server does not permit secure transactions, required by store checkout.
SERIAL_NUM_MATCH_MULTI_ITEMS	{1} different items match this serial number. Select an item from the item dropdown list.
SESSION_TERMD_2ND_LOGIN_DECTD	You can have a maximum of {1} active users at a time in {2}. If you would like to add active users, please contact your account manager to discuss your upgrade options. Or, you may choose to inactivate an existing user before adding a new one. Note that Employee Center users are not included in this total
SESSION_TERMD_2ND_LOGIN_DECTD	You can have a maximum of {1} active users at a time that are enabled for Offline Sales Client. If you would like to add active users, please contact your account manager to discuss your upgrade options. Or, you may choose to disable Offline Sales Client on an existing user before enabling a new one.
SESSION_TIMED_OUT	Your connection has timed out. Please log in again.
SESSION_TIMED_OUT	Your connection has timed out. Please <a href='/pages/login.jsp' target='_self'>log in</a> again.
SESSION_TIMED_OUT	Your session has timed out. Please re-enter your information and try again.
SESSION_TIMED_OUT	Your session has timed out. Please re-print the document(s).
SET_SHIPPING_PICKUP_TYP	Please verify that you have selected a pickup type under setup shipping.
SETUP_METER_REQD	Please set up this meter
SHIP_ADDR_REQD	Shipping address is incomplete.
SHIP_MANIFEST_ALREADY_PRCSSD	A Shipping Manifest has already been processed for the requested date/time {1}.
SHIP_MANIFEST_ERROR	No Shipments found to generate a Shipping Manifest for close date {1} for meter {2}.
SHIP_MANIFEST_ERROR	No Shipping Manifest files found in FedEx Directory for report only request for meter {1}.
SHIP_SETUP_REQD	No {1} registration was found for the location selected. Please select a different shipping item, or go to Setup > Set Up Shipping to register a {2} account for this location.
SHIP_TALBE_UNBALNCD	The Shipping Table is not balanced. Please review the table and ensure there is a Charge for every Range Value, and that there are no duplicates

Error Code Returned	Long Description or Message
SHIPFROM_ADDRESS_NOT_SET	A shipping label could not be generated because the Phone Number of the "Ship From" address is not set. Please go to $(regex) to enter the "Ship From" Phone Number.
SHIPMNT_INSURANCE_NOT_AVAILABLE	Insurance is not available when shipping to the destination country: {1}
SINGLE_VALUE_REQD	Multiple values found for a dropdown field that can only take one.
SITE_DOMAIN_NAME_REQD	Notice: URL Components cannot be used until you have established a domain name for your site
SITE_TAG_ALREADY_EXISTS	This site tag already exists
SITEMAP_GEN_ERROR	An error occurred while trying to generate the Sitemap.
SO_HAS_CHILD_TRANS	This salesOrder has a one or more child transactions associated with it, and cannot be updated.
SO_LINE_HAS_PO	Error: A Drop Ship/Special Order already exists for sales order {1}, line {2}.
SRVC_UNAVAILBL_FOR_LOC	The requested service is unavailable between the selected locations.
SSS_AUTHOR_MUST_BE_EMPLOYEE	The author internal id or email must match an employee.
SSS_CONNECTION_TIME_OUT	The host you are trying to connect to is not responding.
SSS_DEBUG_DISALLWD	Script Debugging is not allowed on this server.
SSS_DRIP_EMAIL_RAN_OUT_OF_COUPON_CODES	Campaign event ran out of coupon codes.
SSS_DUP_DRIP_EMAIL	You are attempting to send the same campaign email twice to the same recipient.
SSS_FILE_SIZE_EXCEEDED	The file you are trying to load exceeds the maximum allowed file size of {1} megabyte.
SSS_INSTRUCTION_COUNT_EXCEEDED	Script Execution Instruction Count Exceeded.
SSS_INVALID_ATTACH_RECORD_TYPE	Attaching of record type {1} to {2} is not supported.
SSS_INVALID_BCC_EMAIL	One or more bcc emails are not valid.
SSS_INVALID_CC_EMAIL	One or more cc emails are not valid.
SSS_INVALID_CMPGN_EVENT_ID	That campaign event is invalid, disabled, or is not a Lead Nurturing Email event. Please select an active Lead Nurturing Email campaign event.
SSS_INVALID_EMAIL_TEMPLATE	That email template is invalid, disabled, or no longer exists. Please select an active email template.
SSS_INVALID_FORM_ELEMENT_NAME	You have entered an invalid form element name. It must be prefixed with "custpage", unique, and cannot contain any non-alphanumeric characters to be added to the form or sublist.
SSS_GSO_FLTR_OPRTOR	SSS Invalid GetSelectOption Filter Operator {1}, must be one of the following: {2}
SSS_INVALID_HEADER	One or more headers are not valid.

Error Code Returned	Long Description or Message
SSS_INVALID_HOST_CERT	An untrusted, unsupported, or invalid certificate was found for this host.
SSS_INVALID_LIST_COLUMN_NAME	You have entered an invalid list column name. It must be unique and cannot contain any non-alphanumeric characters.
SSS_INVALID_LOCK_WAIT_TIME	You have entered an invalid wait time ({1}) for acquiring a lock. The wait time must be greater than 0ms and less than 5000ms.
SSS_INVALID_LOG_TYPE	Execution log type must be one of AUDIT, DEBUG, ERROR, or EMERGENCY.
SSS_INVALID_PORTLET_INTERVAL	You have entered an invalid refresh interval of {1} seconds. It must not be negative.
SSS_INVALID_PORTLET_INTERVAL	You have entered an invalid refresh interval of {1} seconds. It must be at least {2} seconds if not in testing status.
SSS_INVALID_SCRIPTLET_ID	That Suitelet is invalid, disabled, or no longer exists.
SSS_INVALID_SRCH_COL	An nlobjSearchColumn contains an invalid column, or is not in proper syntax: {1}.
SSS_INVALID_SRCH_COLUMN_JOIN	An nlobjSearchColumn contains an invalid column join ID, or is not in proper syntax: {1}.
SSS_INVALID_SRCH_COLUMN_SUM	An nlobjSearchColumn contains an invalid column summary type, or is not in proper syntax: {1}.
SSS_INVALID_SRCH_FILTER	An nlobjSearchFilter contains invalid search criteria: {1}.
SSS_INVALID_SRCH_FILTER_JOIN	An nlobjSearchFilter contains an invalid join ID, or is not in proper syntax: {1}.
SSS_INVALID_SRCH_OPERATOR	An nlobjSearchFilter contains an invalid operator, or is not in proper syntax: {1}.
SSS_INVALID_SUBLIST_OPERATION	You have attempted an invalid sublist or line item operation. You are either trying to cannot access a field on a non-existent line or you are trying to add or remove lines from a static sublist.
SSS_INVALID_SUBMIT_OPTION	You have entered an invalid submit option for this record type: {1}
SSS_INVALID_TYPE_ARG	You have entered an invalid type argument: {1}
SSS_INVALID_UI_OBJECT_TYPE	That operation is not supported for this type of UI object: {1}. It is only supported for type: {2}.
SSS_INVALID_URL	The URL must be a fully qualified HTTP or HTTPS URL if it is referencing a non-NetSuite resource.
SSS_INVALID_URL_CATEGORY	The URL category must be one of RECORD, TASKLINK or SUITELET.
SSS_INVALID_WF_RCRD_TYPE	You have entered an invalid record type {1}. Workflow automation is not supported for this type of record.
SSS_MEMORY_USAGE_EXCEEDED	Script Out Of Memory.
SSS_MISSING_REQD_ARGUMENT	{1}: Missing a required argument: {2}
SSS_QUEUE_LIMIT_EXCEEDED	Script Queue Usage Limit Exceeded

Error Code Returned	Long Description or Message
SSS_RECORD_TYPE_ MISMATCH	The record you are attempting to load has a different type: {1} from the type specified: {2}.
SSS_REQUEST_LIMIT_ EXCEEDED	Request Limit Exceeded!
SSS_REQUEST_TIME_ EXCEEDED	The host you are trying to connect to is not responding or has exceeded the maximum allowed response time.
SSS_SCRIPT_DESERIA LIZATION_FAILURE	Scheduled Script deserialization failure
SSS_SEARCH_TIMEOUT	Your search request has timed out. You may need to refine your search or combine the results of multiple searches to achieve the desired result.
SSS_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.
SSS_STACK_FRAME_DEPTH_ EXCEEDED	Script Stack Frame Depth Exceeded
SSS_TIME_LIMIT_EXCEEDED	Script Execution Time Exceeded.
SSS_TRANS_IN_PROGRESS	You cannot call nlapiBeginTransaction() because there is already a transaction in progress.
SSS_TRANSACTION_REQD	That operation can only be performed when there is a transaction in progress
SSS_UNKNOWN_HOST	The host you requested {1} is unknown or cannot be found.
SSS_USAGE_LIMIT_ EXCEEDED	Script Execution Usage Limit Exceeded
START_DATE_AFTER_END_ DATE	The start date must preceed the end date.
START_DATE_REQD	Please enter a value for {1} Start Date
STATE_ALREADY_EXISTS	A State/Province/County with the same name already exists.
STATE_REQD	State is a required field and it cannot be null or empty.
STATUS_ASSIGNEE_REQD	The issue status {1} does not define an assignee issue role. That status may not be used until this is corrected.
STORAGE_LIMIT_EXCEEDED	You entered a value that will exceed the internal storage limit of {1}. Please reduce the number.
STORE_ALIAS_UNAVAILABLE	The Store alias you chose "{1}" is already taken. Please go back and choose another.
STORE_DOMAIN_ UNAVAILABLE	The store domain name you chose '{1}' is already taken. Please go back and choose another.
SUB_MISMATCH	Resources for Time and Material projects must be in the customer's subsidiary. The following resources are in a different subsidiary: {1}
SUB_RESTRICT_VIEW_REQD	Subsidiary Restrict View Required: You must first restrict your view to a subsidiary before running this report (Home > Set Preferences ><a href="/app/center/userprefs.nl?whence={1}">Restrict View</a>).
SUB_TAX_AGENCY_REQD	No tax agency defined for subsidiary
SUBITEM_REQD	You must first select the new subitems on the matrix tab you want to add.

Error Code Returned	Long Description or Message
SUBITEM_REQD	You must first select the subitems on the matrix tab you want to create.
SUBSIDIARY_MISMATCH	The employee and billable customer must be in the same subsidiary.
SUCCESS_TRANS	The transaction was entered {1} successfully, {2}
SUPRT_CNTR_LOGIN_ERROR	{1} Support Center login error: we are unable to find the customer record for account={2}
TAG_ALREADY_EXISTS	That tag name is already being used. Please go back and rename it.
TAG_SUBSTITUTN_ERROR	Error: Tag substitution loop encountered. Tag substitution sequence: {1}
TAG_SUBSTITUTN_ERROR	Error: Tag substitution result is too large. Tag substitution sequence: {1}
TAGATA_ALREADY_ENDORSED	The Receivable Tegata has already been endorsed.
TAX_ACCT_SETUP_REQD	Tax Accounts Not Defined.
TAX_CODE_REQD	No default tax code is defined for country {1}
TAX_CODES_SETUP_PROBLEM	The tax codes haven't been set properly
TAX_CODES_SETUP_REQD	Can't open store for {1}. This company does not have its tax codes fully set up. This is required to properly calculate taxes on international, other-province and same-province orders.
TAX_CODES_SETUP_REQD	The company is not usable. Administrator hasn't set up the tax codes.
TAX_GROUP_SETUP_REQD	You have not created tax groups in your NetSuite account. To ensure that your customers are charged the correct amount of sales tax, you must create tax groups by entering them manually at Lists > Accounting > Tax Groups > New.
TAX_PRD_REQD	No Current Tax Period is defined. <a href='/app/setup/period/taxperiods.nl'>Click here</a> to create a tax period.
TAX_SETUP_REQD	The tax period range {1} has not been defined. Please visit '<A href='{2}'>Setup > Accounting > Manage Tax Periods</A>' to define this period or set up your year.
TEMPLATE_NOT_FOUND	Template not found
TEMPLATE_NOT_FOUND	Template Record not found
THIRD_PARTY_BILLING_ACCT_REQD	A 3rd Party Billing Account Number must be provided when selecting a 3rd Party Billing Type.
TICKET_NOT_LOCATED	The ticket {1} cannot be located in the error database. If this is from a customer logged case, the error may not yet be inserted into the system.
TIME_ENTRY_DISALLWD	{1} does not allow time entry.
TIMEOUT_THE_RECORD_DOESNT_EXIST_ANYMORE	Timeout: the record doesn't exist anymore.
TOPIC_REQD	You must select and add a topic to this solution.
TRAN_DATE_REQD	Missing transaction date.
TRAN_LINE_FX_AMT_REQD	Missing foreign currency amount on non-variance transaction line
TRAN_LINK_FX_AMT_REQD	Missing foreign currency amount on non-variance transaction link

Error Code Returned	Long Description or Message
TRAN_PERIOD_CLOSED	The action is causing generation of foreign exchange variance in a closed period. Please retry with {1} as the exchange rate.
TRAN_PERIOD_CLOSED	You are not allowed to change the revenue recognition status for one or more lines on this transaction as it would impact a closed period.
TRAN_PERIOD_CLOSED	You cannot change the G/L impact of a transaction in a closed period.
TRAN_PRD_CLOSED	This action cannot be completed because it requires modification of the transaction in a closed period due to foreign exchange variance. You may either open the period for this transaction or use the same rate ({1})between the transactions that will be linked.
TRANS_ALREADY_ REFUNDED	Transaction Already refunded. A refund has already been performed on the transaction.
TRANS_ALREADY_SETTLED	Transaction Already Settled. Void failed because transaction has already settled, submit credit.
TRANS_ALREADY_VOIDED	Transaction Already Voided. Void failed because transaction is already voided.
TRANS_AMTS_UNBALNCD	Transaction is not in balance! amounts+taxes+shipping: {1}, total amount: {2}
TRANS_APPLIED_AMTS_ UNBALNCD	Transaction is not in balance! Total to apply of ${1} does not equal sum of applied ${2} and unapplied ${3}
TRANS_APPLIED_AMTS_ UNBALNCD	Transaction is not in balance! Total to apply of ${1} does not equal sum of payment ${2} and credits ${3} and deposits ${4}
TRANS_CLASS_UNBALNCD	Transaction out of balance for class {1} total = {2}.
TRANS_DEPT_UNBALNCD	Transaction out of balance for department {1} total = {2}.
TRANS_DOES_NOT_EXIST	No transaction exists for that entity.
TRANS_DSNT_EXIST	The transaction you are attempting to access does not exist.
TRANS_EDIT_DISALLWD	This transaction is in a period that has been closed. You may not edit it.
TRANS_EDIT_DISALLWD	You cannot edit this transaction. {1} does not support the imported transaction.
TRANS_FORGN_CRNCY_ MISMATCH	Transaction and foreign currency account use different currencies.
TRANS_FORGN_CUR_ UNBALNCD	Transaction was not in balance (Foreign currency). Posting total = {1}
TRANS_FORGN_CUR_ UNBALNCD	Transaction was not in balance (Foreign currency). Total = {1}
TRANS_IN_USE	This transaction cannot be deleted because it is linked to by one or more transactions. Click <a href='/app/accounting/transactions/payments.nl?id={1}&label={2}&type={3}&alllinks=T'>here>/a> to see the list of linked transactions.
TRANS_LINE_AND_PMT_ UNBALNCD	Transaction is not in balance! Line item sum of ${1} not equal to payment amount ${2}
TRANS_LINES_UNBALNCD	Transaction is not in balance! Line item sum of ${1} does not equal amount of ${2}
TRANS_LINES_UNBALNCD	Transaction is not in balance! Line item sum of ${1} does not equal applied amount of ${2}

Error Code Returned	Long Description or Message
TRANS_LOC_UNBALNCD	Transaction out of balance for location {1} total = {2}.
TRANS_NOT_CLEANED	Transaction not cleaned up.
TRANS_NOT_COMPLETED	Transaction was not complete.
TRANS_PRCSSNG_ERROR	Errors occurred while processing the selected transaction. Please process it individually for more information.
TRANS_UNBALNCD	The debits and credits are not balanced on this transaction because amounts entered include more decimal places than are supported for this currency. Please round off the amount of each line to a maximum of {1} decimal places.
TRANS_UNBALNCD	Transaction is not in balance! {1}
TRANS_UNBALNCD	Transaction is not in balance! {1},{2} othercount = {3}
TRANS_UNBALNCD	Transaction out of balance for {1} {2} total = {3}.
TRANS_UNBALNCD	Transaction was not in balance. Posting total = {1}
TRANS_UNBALNCD	Transaction was not in balance. Total = {1}
TRANSACTION_DELETED	The transaction you are attempting to access has been deleted.
TRANSORD_SHIP_REC_ MISMATCH	You can not recieve more from a transfer order than you have shipped
TWO_FA_AUTH_REQD	All your other roles require a one-time key at login. Please click "Go Back" to enter a one-time key or contact the company's administrator if you have questions.
TWO_FA_REQD	Two-Factor Authentication required
UNABLE_TO_PRINT_CHECKS	Unable to print checks.
UNABLE_TO_PRINT_ DEPOSITS	Unable to print deposits.
UNAUTH_CAMPAIGN_ RSPNS_RQST	Unauthorized campaign response request
UNAUTH_UNSUBSCRIBE_ RQST	Unauthorized unsubscribe request
UNDEFINED_ACCTNG_PRD	The accounting period range {1} has not been defined. Please visit '<A href='/app/setup/period/fiscalperiods.nl'>Setup > Accounting > Manage Accounting Periods</A>' to define this period or set up your year.
UNDEFINED_ACCTNG_PRD	The comparison accounting period range {1} has not been defined. Please visit '<A href='/app/setup/period/fiscalperiods.nl'>Setup > Accounting > Manage Accounting Periods</A>' to define this period or set up your year.
UNDEFINED_ACCTNG_PRD	The default accounting period for this report has not been defined. Please visit '<A href='/app/setup/period/fiscalperiods.nl'>Setup > Accounting > Manage Accounting Periods</A>' to define this period or set up your year.
UNDEFINED_CSTM_FIELD	Undefined customfield.
UNDEFINED_TAX_PRD	The default tax period for this report has not been defined. Please visit '<A href='/app/setup/period/taxperiods.nl'>Setup > Accounting > Manage Tax Periods</A>' to define this period or set up your year.

Error Code Returned	Long Description or Message
UNEXPECTED_ERROR	An error occurred while processing item options.
UNEXPECTED_ERROR	An unexpected error has occurred.
UNEXPECTED_ERROR	An unexpected error has occurred. A FedEx Shipping Label was not generated.
UNEXPECTED_ERROR	An unexpected error has occurred while generating this content.<p>Our Customer Support staff have been notified and are looking into the problem.
UNEXPECTED_ERROR	An unexpected error has occurred while synching a record. Click [OK] to skip the record and continue.
UNEXPECTED_ERROR	An unexpected error has occurred. Technical Support has been alerted to this problem.
UNEXPECTED_ERROR	An unexpected error occurred while extracting email from SMTP server
UNEXPECTED_ERROR	An unexpected error occurred while logging email request completion
UNEXPECTED_ERROR	An unexpected error occurred while logging email request failure
UNEXPECTED_ERROR	An unexpected error occurred while logging email request start
UNEXPECTED_ERROR	An unexpected error occurred while processing the payment.
UNEXPECTED_ERROR	An unexpected error occurred with the group SQL
UNEXPECTED_ERROR	An Unexpected JavaScript Error has occurred
UNEXPECTED_ERROR	Error
UNEXPECTED_ERROR	Error: {1}
UNEXPECTED_ERROR	Please specify an scompid
UNEXPECTED_ERROR	Problem during commission calculation
UNEXPECTED_ERROR	An unexpected error occurred.
UNEXPECTED_ERROR	Dto java class is not defined for {1}.
UNEXPECTED_ERROR	Server error: no dto class is defined for record of type {1}
UNEXPECTED_ERROR	Server error: missing database entries in WSRecordElement and WSNameSpace table for object of {1}
UNEXPECTED_ERROR	Application error: no form request class is defined for record of type {1}
UNIQUE_CONTACT_NAME_REQD	. Contact names must be unique
UNIQUE_CUST_EMAIL_REQD	A customer record with this email address already exists. You must enter a unique customer email address for each record you create.
UNIQUE_CUST_EMAIL_REQD	A customer record with this email address already exists. You must enter a unique customer email address for each record you create. To correct this record, click <a href='javascript:history.go(-1);';>back</a> and enter a new customer email address in the Customer field. Then, click Submit.
UNIQUE_CUST_ID_REQD	A customer record with this ID already exists. You must enter a unique customer ID for each record you create.
UNIQUE_CUST_ID_REQD	A customer record with this ID already exists. You must enter a unique customer ID for each record you create. To correct this record, click <a

Error Code Returned	Long Description or Message
	href='javascript:history.go(-1);';>back</a> and enter a new customer ID in the Customer field. Then, click Submit.
UNIQUE_ENTITY_NAME_ REQD	multiple sub-customers or jobs have name '{1}' which would create a naming conflict upon merge. All names must be unique. Before merging, you must change one of the subs named '{2}' to something else.
UNIQUE_GROUPID_REQD	You must specify exactly one numeric groupId
UNIQUE_PARTNER_CODE_ REQD	{1:name of partner record} Code "{2:partner code}" already exists. Please select a unique code for each record.
UNIQUE_QTY_REQD	Quantities must be unique
UNIQUE_RCRD_ID_REQD	A record with this ID already exists. You must enter a unique ID to create or update this record.
UNIQUE_SOLUTION_CODE_ REQD	A solution with this particular solution code already exists. Please assign a different code.
UNITS_TYP_IN_USE	This units type is used by {1} {2}. You must delete the {2} and all associated transactions to delete this units type.
UNKNOWN_CARRIER	Package Tracking is not available for id {1}. Unknown carrier.
UNKNOWN_RCRD_TYPE	Unknown record type
UNKNOWN_SCRIPT_TYP	Unknown Script Type
UNKNWN_ALLOCTN_ SCHDUL_FREQ_TYP	Unable to determine allocation schedule frequency type.
UNKNWN_EMAIL_AUTHOR	The author of this email cannot be found.
UNKNWN_EXCHANGE_RATE	Unable to determine the exchange rate for currency symbol {1}. Please create a currency item with an exchange rate for this currency.
UNRECOGNIZED_METHOD	unrecognized method '{1}'
UNSUBSCRIBE_REQD	Unsubscribe is mandatory, please enter a value for this field.
UNSUPPORTED_METHOD	Unsupported method
UNSUPPORTED_WS_ VERSION	TBA not supported in this SuiteTalk endpoint version
UNSUPRTD_DOC_TYP	You attempted to upload an unsupported document type. Please try again with a selection from the list below:
UPDATE_DISALLWD	Update is not allowed
UPDATE_PRICE_AMT_REQD	Please specify an amount to update prices.
UPGRADE_WS_VERSION	Could not set '{1}' to field '{2}' of record number {3} due to schema enumeration restriction.
UPGRADE_WS_VERSION	Please consider upgrading to endpoint {1}
UPGRADE_WS_VERSION	Sales order <id {1}> contains item serial/lot numbers that are not supported in your client application. You are not allowed to update serial/lot numbers on this sales order. Contact your software vendor for the latest Web Services upgrade.
UPGRADE_WS_VERSION	Sales order <id {1}> has items with more than one serial/lot numbers that is not supported in your client application. The serial/lot numbers have been removed to successfully return the sales order. Contact your software vendor for the latest Web Services upgrade.

Error Code Returned	Long Description or Message
UPGRADE_WS_VERSION	This {1} has multiple {2}s. Web Services schema version {3} or greater is required to modify {2} for this {1}
UPGRADE_WS_VERSION	{1} {2} has multiple shipping routes enabled, which is only supported in version 2008_2 and newer. The shipping information has been omitted to successfully return this record.
UPS_CANT_INTEGRATE_ FULFILL	The fulfillment cannot be integrated with UPS because the Shipping Integration Carrier is not set to UPS.
UPS_CONFIG_ERROR	A UPS configuration error occured. Please contact tech support.
UPS_LICENSE_AGREEMNT_ REQD	You must agree to the UPS license agreement
UPS_ONLINE_RATE_ UNAVAILBL	The UPS Online Realtime Rates System is temporarily unavailable. Please resubmit your rate request in a few minutes.
UPS_ONLINE_RATE_ UNAVAILBL	UPS did not return any rates for the specified origin and destination addresses.
UPS_ONLINE_SHIP_ UNAVAILBL	The UPS Online Shipping System is temporarily unavailable. Please resubmit your fulfillment in a few minutes.
UPS_REG_NUM_IN_USE	The submitted UPS Registration Number, {1}, is already in use. Please resubmit the registration with a different UPS registration Number.
UPS_SETUP_REQD	No UPS registration was found. Please register your UPS account with NetSuite before attempting to send a fulfillment request to UPS.
UPS_VOID_ERROR	The UPS Void failed due to a system failure.
UPS_XML_ERROR	XML Sent to UPS. UPS returned error code/text:
URL_ID_PARAM_REQD	URL is missing the id parameter. The file could not be retrieved.
URL_REQD	You must enter a URL for this media item.
USER_CONTEXT_REQD	User context is not set
USER_DISABLED	user disabled
USER_ERROR	An error occurred during your last update.
USER_ERROR	A User Error Has Occurred
USER_ERROR	Detach requires an AttachBasicReference
USER_ERROR	Either internalId or externalId is required.
USER_ERROR	Folder cannot be made a subfolder of itself.
USER_ERROR	Gift Certificate From, Recipient Name, and Recipient Email are required.
USER_ERROR	Invalid Attachment record combination
USER_ERROR	Missing Item Weight or Weight Unit.
USER_ERROR	Missing or Invalid RecordType for AttachTo
USER_ERROR	Must submit a non-abstract instance of baseRef (eg RecordRef, CustomRecordRef) NOT a baseRef
USER_ERROR	Must submit a non-abstract instance of record or searchRecord (eg customer or customerSearchBasic).
USER_ERROR	No budget inserted or updated.

Error Code Returned	Long Description or Message
USER_ERROR	{1}
USPS_ACCT_NUM_ALREADY_EXISTS	There is an existing NetSuite registration for Endicia account number {1}.
USPS_INVALID_INSURED_VALUE	Insured value exceeds the {1} maximum allowed by Endicia.
USPS_INVALID_PACKAGING	The Carrier Packaging that you have selected is not valid for this item fulfillment. Usually this indicates the selected packaging cannot be used with the selected USPS shipping method, or the package weight is invalid. Please check the documentation for more details.
USPS_INVALID_PSWD	The Endicia Web Password does not match the Web Password for this USPS Registration account number.
USPS_LABEL_VOIDED	This error required 1 or more labels created for this transaction to be voided at Endicia.
USPS_LABEL_VOIDED	This error required 1 or more labels created for this transaction to be voided at Endicia.
USPS_MAX_ITEM_EXCEEDED	International USPS fulfillments allow a maximum of 5 unique items per package, due to customs documentation. If more than one package is required, please break up the shipment into multiple fulfillments.
USPS_ONE_PACKAGE_ALLWD	International USPS fulfillments allow only one package. If more than one package is required, please break up the shipment into multiple fulfillments of one package each.
USPS_PASS_PHRASE_NOT_UPDATED	The Endica Pass Phrase was not updated: {1}
USPS_REFUND_FAILED	Failed Endicia Refund Request
USPS_REFUND_FAILED	The Endicia Refund Request failed due to a system error.
USPS_RETRY	A response was not received for the USPS funding request. Please try again in a few minutes.
USPS_VALIDATE_ADDR	The address you entered could not be validated. Please verify the city, state, and/or zip code. You can validate an address by visiting the <a href="http://zip4.usps.com/zip4/welcome.jsp" target="_blank">U.S. Postal Service</a> web site, or the <a href="http://www.endicia.com/Developers/ZipLookup/" target="_blank">Endicia</a> web site.
USPS_VERIFY_TRACKING_NUM	Please verify that the following tracking numbers were created and voided in your Endicia account before proceeding.
USPS_VOID_ERROR	An error was detected during the Endicia Void operation:
USPS_VOID_ERROR	An error was detected during the Endicia Void operation:
VALID_EMAIL_REQD	Missing or invalid email address. Email address is a required field and it cannot be null or empty. The email address must be in a valid format.
VALID_EMAIL_REQD_FOR_LOGIN	Please enter a valid email address when granting login access privileges to this record.
VALID_FIRST_NAME_REQD	Missing or invalid First Name. Users first name is a required field and cannot be null or empty.
VALID_LAST_NAME_REQD	Missing or invalid Last Name. Users last name is a required field and cannot be null or empty.

Error Code Returned	Long Description or Message
VALID_LINE_ITEM_REQD	You must have at least one valid line item for this transaction.
VALID_PHONE_NUM_REQD	Missing or invalid Home phone number. The Home phone number is a required field and it cannot be null or empty. The format of the Home phone number must contain area code plus seven digit number.
VALID_PRD_REQD	Insert Transaction Failure: No valid, open, posting period for date - {1}. Please visit Setup > Manage Accounting Periods to set up a new accounting period.
VALID_PRD_REQD	Insert Transaction Failure: No valid, open, tax period for date - {1}. Please visit Setup > Manage Tax Periods to set up a new tax period.
VALID_PRD_REQD	Update Transaction Failure: No valid, open, {1} period for date - {2}
VALID_URL_REQD	Please go back and provide a valid URL for all five fields on the External tab.
VALID_VERSION_REQD_IN_URL	If the version parameter is passed through the URL, it MUST contain a valid version in a phased release environment. Valid: {1}
VALID_WORK_PHONE_REQD	Missing or invalid Work phone number. The Work phone number is a required field and it cannot be null or empty. The format of the Work phone number must contain area code plus seven digit number.
VALID_ZIPCODE_REQD	Missing or invalid ZIP code field. ZIP code is a required field and it cannot be null or empty. ZIP code and state values are checked against an internal database to make sure that ZIP code specified exists in state specified.
VENDOR_TYPE_REQD	No Vendor Type was specified. If creating a Tax Agency, please ensure that the vendor type is active and marked as a tax agency.
VERIFY_DESTNTN_ZIP_CODE	Please verify that the destination zipcode is correctly specified.
VERIFY_PAYROLL_FUND_ACCT	The payroll funding account has not been verified. Please verify the payroll funding account
VERIFY_ZIP_CODE_SETUP	Please verify that you have correctly set your zip code under setup company. If you have multi-location enabled, verify that you have set a correct zipcode for each location.
VISA_ERROR	Communication error with Visa. Please retry.
VOID_FAILED	Void Failed. Failed to void transaction, retry void or issue credit.
VOIDING_REVERSAL_DISALLWD	You may not create a voiding reversal for transactions with inventory impact. To reverse the inventory impact of the transaction, you will need to create an inventory adjustment.
VSOE_CANT_ADD_ITEM_GROUP	When the <b>Is VSOE bundle</b> box is checked, Items for Purchase cannot be added to item groups.
VSOE_REV_REC_TMPLT_REQD	All Lines in a VSOE Bundle with a VSOE Allocation must have a revenue recognition template.
VSOE_TOTAL_ALLOCATION_ERROR	The total vsoe allocation in a bundle must equal the total bundle sales amount.
VSOE_TRAN_VSOE_BUNDLE_ERROR	You have indicated that you would like this transaction to be treated as a Bundle (multi-element arrangement) for VSOE purposes. Please either uncheck the 'Transaction Is VSOE Bundle' checkbox or remove the Item Groups that have the 'Is VSOE Bundle' option specified.

Error Code Returned	Long Description or Message
WF_EXEC_USAGE_LIMIT_ EXCEEDED	Workflow Execution Usage Limit Exceeded
WORK_DAYS_REQD	Select one or more working days.
WORLDPAY_ERROR	A failure occurred while attempting to connect to WorldPay.
WORLDPAY_ERROR	There was a problem with your WorldPay credentials. Please be sure you are using the correct combination of Installation ID, Merchant Code and XML Password
WRITE_OFF_ACCT_REQD	To receive items without restocking, you must first set a value for the write-off account. To set the value of the write-off account, go to Accounting > Accounting Preferences > Order Management > Write-Off Account for Returns.
WS_CONCUR_SESSION_ DISALLWD	Someone has logged in as this user from a different web services session. Only one person may login as a particular user at a time. As a consequence, this session has been terminated.
WS_CONCUR_SESSION_ DISALLWD	Only one request may be made against a session at a time
WS_EXCEEDED_CONCUR_ USERS_ALLWD	You can have a maximum of {1} active concurrent WS users at a time in {2}
WS_FEATURE_REQD	You have not enabled web services feature for your account.
WS_INVALID_SEARCH_ OPERATN	When using request-level credentials, you must use the {1} operation instead of {2}
WS_LOG_IN_REQD	You must log in before performing a web service operation.
WS_PERMISSION_REQD	You do not have permission to access web services feature.
WS_REQUEST_BLOCKED	SuiteTalk concurrent request limit exceeded. Request blocked.
ZIP_FILE_CONTAINS_VIRUS	The zip file contains a virus {1}. Upload abort.

Warning Status Codes

The code USER_ERROR is returned in the code field of the statusDetail type where the type attribute has a value of warning. A warning message is also returned which varies depending on the exact cause of the warning and is usually self-descriptive. For a complete description of warnings and how they differ from errors and faults, refer to Web Services Warnings, Errors, and Faults.