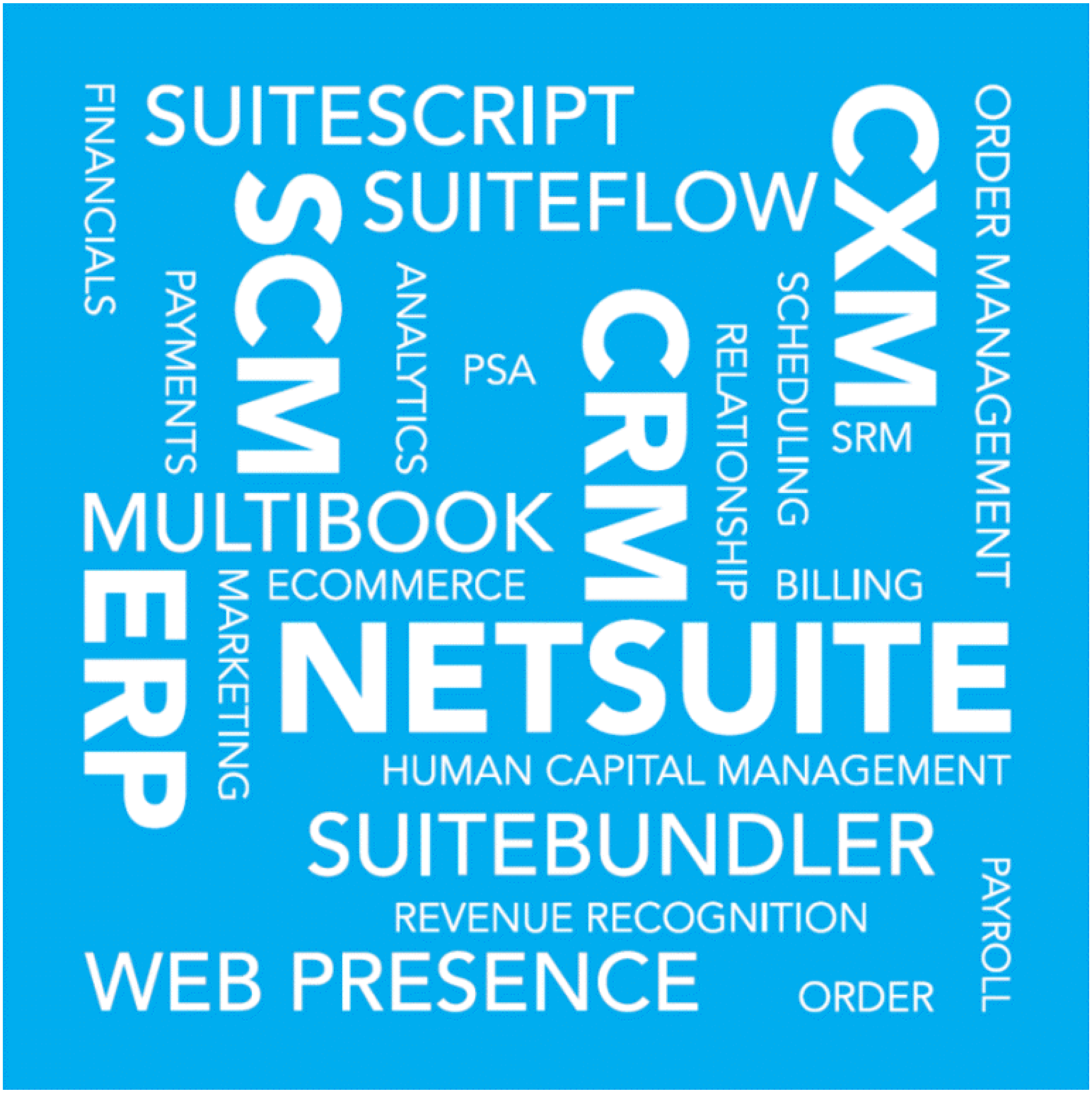
SuiteTalk (Web Services) Records Guide



December 13, 2017 Version 2017 Release 2

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Sample Code

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

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

No Excessive Use of the Service

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

Beta Features

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

Table of Contents

SuiteTalk Records Overview

In this Guide

This manual contains the following information:

- All records supported in SuiteTalk
- A list of all supported operations for each record
- Usage notes or sample code to show how to work with specific records
- Within each record description, links to the SuiteTalk Schema Browser for all field definitions and field level help
- Guidelines for using the SuiteTalk Schema Browser, which provides up-to-date reference information for each record supported in SuiteTalk

Important: For information on SuiteTalk platform features, operations, types, warnings, errors, and faults, see the SuiteTalk (Web Services) Platform Guide in the NetSuite Help Center. Note that the material in the SuiteTalk (Web Services) Records Guide currently pertains to the **2017.2** endpoint.

How to Use Web Services Records Help

The Records Guide must be used in conjunction with the SuiteTalk Schema Browser. Each record listed in this guide includes a link to the SuiteTalk Schema Browser. Within the Schema Browser you will find all available fields, sublists, and search filter fields for that record. The Schema Browser also provides field level help for each field that appears on the record.

Within this guide, the description of each record generally includes the following types of information:

- **Supported operations** – a list of all operations that can be used with the record.
- **Field definitions** – provides a link to the Schema Browser and, in some cases, additional explanations about how to work with particular fields.
- **Usage notes** – additional details or context about working with the record.
- **Code samples** – examples of how to structure your code, usually in either Java or C#, with the accompanying SOAP request and response.

Important: For a list of all NetSuite records that are supported in SuiteTalk, see Web Services Supported Records.

Working Online

If you are working online, using either the PDF or online help version of this guide, click the links to the SuiteTalk Schema Browser that are provided within each record description. The Schema Browser will open in your default browser.

Downloading the SuiteTalk Schema Browser

A .zip file of the Schema Browser is located on the NetSuite Developer Portal. Navigate to http://www.netsuite.com/portal/developers/resources/suitetalk-documentation.shtml, and locate the link titled View v2017.2 SuiteTalk Schema Browser Zipped. After you download the .zip and extract all files, navigate to the schema directory. To view the Schema Browser content, open index.html file in the browser of your choice.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Web Services Supported Records

The following records are currently supported in SuiteTalk.

Entities

- Contact
- Customer
- Customer Status
- Employee
- Entity Search
- Group (Entity Group)
- HCM Job
- Partner
- Project (Job)
- Project Status (Job Status)
- Project Type (Job Type)
- Vendor

Activities

- Events (CalendarEvent)
- Phone Call
- Project Task
- Resource Allocation
- Tasks

Marketing

- Campaign
- Campaign Audience
- Campaign Category
- Campaign Channel
- Campaign Family
- Campaign Offer
- Campaign Response
- Campaign Search Engine

- Campaign Subscription
- Campaign Vertical
- Coupon Code
- Promotion Code

Transactions

- Advanced Intercompany Journal Entry
- Assembly Build
- Assembly Unbuild
- Bin Putaway Worksheet
- Bin Transfer
- Budget
- Cash Refund
- Cash Sale
- Charge
- Check
- Credit Memo
- Customer Deposit
- Customer Payment
- Customer Refund
- Deposit
- Deposit Application
- Estimate/Quote
- Expense Report
- Inbound Shipment
- Intercompany Journal Entry
- Intercompany Transfer Order
- Inventory Adjustment
- Inventory Cost Revaluation
- Inventory Transfer
- Invoice
- Item Demand Plan
- Item Fulfillment
- Item Receipt
- Item Supply Plan
- Journal Entry
- Manufacturing Operation Task
- Opportunity
- Paycheck
- Paycheck Journal
- Purchase Order

- Requisition
- Return Authorization
- Sales Order
- Statistical Journal Entry
- Time Bill (Track Time)
- Transfer Order
- Usage
- Vendor Bill
- Vendor Credit
- Vendor Payment
- Vendor Return Authorization
- Work Order
- Work Order Issue
- Work Order Close
- Work Order Completion
- Multiple Shipping Routes in Web Services

Support

- Issue
- Support Case
- Support Case Issue
- Support Case Origin
- Support Case Priority
- Support Case Status
- Support Case Type
- Solution
- Topic

File Cabinet

- File
- Folder

Items

- Assembly Item (BOM Item)
- Description Item
- Discount Item
- Download Item
- Gift Certificate Item
- Inventory Item
- Item Group

- Kit/Package Item
- Lot Numbered Assembly Item
- Lot Numbered Inventory Item
- Markup Item
- Noninventory Purchase Item
- Noninventory Resale Item
- NonInventory Sales Item
- Other Charge Purchase Item
- Other Charge Resale Item
- Other Charge Sale Item
- Payment Item
- Serialized Assembly Item
- Serialized Inventory Item
- Service Purchase Item
- Service Resale Item
- Service Sale Item
- Subtotal Item

Communication

- Message
- Note

Website

- Site Category

Lists

- Account
- Accounting Period
- Billing Account
- Billing Schedule
- Bin
- Classification (Class)
- Consolidated Exchange Rate
- Currency
- Currency Rate
- Department
- Expense Category
- Fair Value Price
- Gift Certificate
- Inventory Detail

- Inventory Number
- Item Revision
- Location
- Manufacturing Cost Template
- Manufacturing Routing
- Nexus
- Payroll Item
- Revenue Recognition Schedule
- Revenue Recognition Template
- Sales Tax Item
- Subsidiary
- Tax Group
- Tax Type
- Unit Type

Other Lists

- Budget Category
- Contact Category
- Contact Role
- Customer Category
- Customer Message
- Global Account Mapping
- Item Account Mapping
- Lead Source
- Note Type
- Other Name Category
- Partner Category
- Payment Method
- Price Level
- Pricing Group
- Sales Role
- Tax Control Account
- Term
- Vendor Category
- Win Loss Reason

Customization

- Custom Record
- Custom Record Custom Field
- Custom Record Type

- Custom List
- Custom Transaction
- CRM Custom Field
- Entity Custom Field
- Item Custom Field
- Item Option Custom Field
- Other Custom Field
- Transaction Body Custom Field
- Transaction Column Custom Field

Subrecords

- Address
- Inventory Detail
- Item Search

Searches

- Accounting Transaction Search
- Item Search
- Transaction Search

SuiteTalk Schema Browser

Viewing the SuiteTalk Schema Browser

Go to the SuiteTalk Schema Browser.

The SuiteTalk Schema Browser provides a summary of all records, sublists, and other objects available in SuiteTalk. Information about each object is displayed as a series of tables, both for ease of browsing and to provide additional details compared with what is available in the WSDL.

For more details about the Schema Browser, see the following:

- Finding a Record or Subrecord
- Finding Other Objects
- Default View
- Namespace View

Note: Only the 2017.1 Schema Browser is currently supported. Older version may still be accessible, but they are not supported.

Finding a Record or Subrecord

The SuiteTalk Schema Browser is designed to let you browse for records and subrecords by name or by namespace.

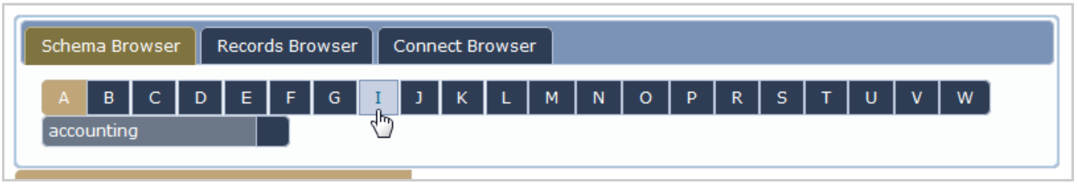
Using the A-Z Index

If you know the name of the record or subrecord you are looking for, use the A-Z index at the top of the browser window.

To find a record or subrecord using the A-Z index:

1. Click the appropriate letter at the top of the browser window.

   The pane at the left updates to include a list of all records whose names begin with the letter you selected. The center pane updates to show details of the first record in the list.



2. In the left-hand pane, find the name of the record or subrecord you are interested in, and click it.

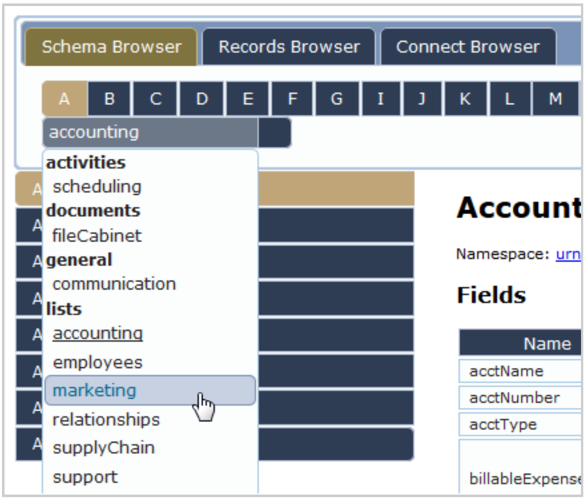
   The center pane updates to show details of the record. For help understanding the details displayed, see Summary of the Record.

Using the Namespace Dropdown List

If you do not know the name of the record, you might want to browse by namespace using the dropdown list, which is located below the A-Z index.

To find a record using the namespace dropdown list:

1. Use the dropdown list to select the appropriate namespace.



   The pane at the left updates to include a tabbed list of all objects in this namespace, with the Record subtab selected. The center panel updates to include details about the first record in the list.

2. In the left-hand pane, find the name of the record or subrecord you are interested in, and click it.

   The center pane updates to show details of this record. For help understanding the details displayed, see Summary of the Record.

Finding Other Objects

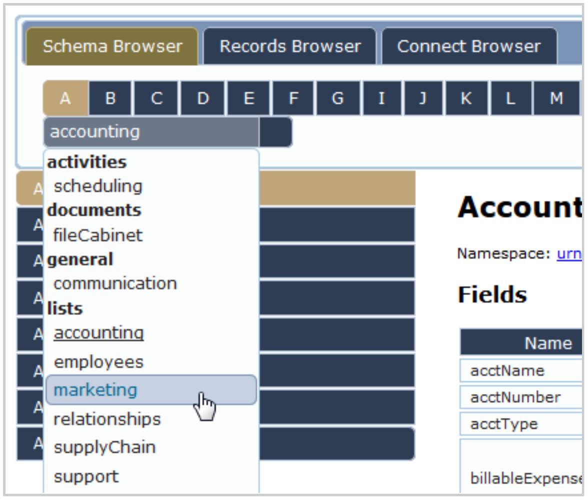
Often when you go the SuiteTalk Schema Browser, you are looking for details about a record. However, you may also want to find details of sublists, search objects, enumerations, and data types such as RecordRef and BaseRef. This topic explains how to find these other objects in the Schema Browser.

Using the Namespace Dropdown List

If you know the namespace that the object belongs to, you can navigate to it using the namespace dropdown list, which is located below the A-Z index.

To find an object using the namespace dropdown list:

1. Use the dropdown list to select the appropriate namespace.



   The pane at the left updates to include a tabbed list of all objects in this namespace, with the Record subtab selected. The center panel updates to include details about the first record in the list.

2. If appropriate, change the view by doing one of the following:
   - To find a search object, click the Search subtab.
   - To find sublists, their child elements, and certain other objects, click the Other subtab.
   - To find an enumeration, click the Enum tab.

   The buttons at the left update according to the choice you made in Step 2.

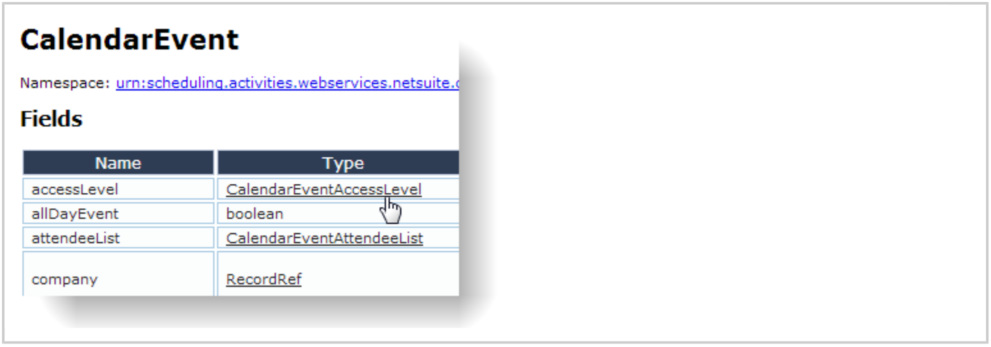
3. In the list at the left, locate the appropriate object, and click it.

Using Other Controls

In some cases, you may be interested in an object directly related to the record you are currently viewing in the Schema Browser. In these cases, you may be able to navigate to that object in a more direct way. For example:

- If you are interested in a search, scroll to the bottom of the page summarizing the record. All search objects directly related to the record are listed here. You can click the name of any search to navigate to a reference page summarizing that search object.

- Many values listed in the Type column of the Fields table are links that take you directly to another reference page in the Schema Browser. That is, the Type column includes sublists, enumerations, and other object types, such as BaseRef and RecordRef. You can click any of these to view the reference page for that object.



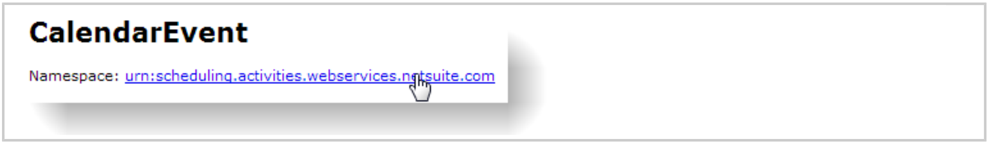
CalendarEvent

Namespace: urn:scheduling.activities.webservices.netsuite.c

Fields

Name	Type
accessLevel	CalendarEventAccessLevel
allDayEvent	boolean
attendeeList	CalendarEventAttendeeList
company	RecordRef

- If you want to browse objects in the same namespace as the record you are viewing, click the URL below the name of the record. The page automatically updates to include, in the pane at the left, a tabbed list of all objects in the namespace. See also Navigating to a Different Object.



CalendarEvent

Namespace: urn:scheduling.activities.webservices.netsuite.com

Default View

In the default view of the SuiteTalk Schema Browser, the pane at the left side of the screen lists all the records and subrecords whose names begin with the letter you selected in the A-Z index, whereas the center pane describes details of one record. This topic describes a few highlights of the information you see in this default view.

Summary of the Record

The center pane of the default view is essentially a summary of information about one record.

Namespace Link

At the top of the page, directly below the name of the record, is a link to the namespace to which the record belongs.

If you click this link, the pane at the left refreshes to show a tabbed list of all objects in the namespace.

List of Fields

Data about the record’s fields is displayed in a table. This table includes the following columns:

- **Name** — The name of the field as it exists in web services.
- **Type** — The data type of the field. If the name of the type is underlined, you can click it to view a reference page about the type. (When you do, the pane at the left updates to include a tabbed list of other objects from the namespace.)

- **Cardinality** — A measure that shows two values representing the “minimum occurs” and “maximum occurs” attributes as defined for this record in the corresponding XSD file. For example, a value of 0..1 means that the field is not required and that it can be populated a maximum of one time. A value of 0..unbounded means that the field is not required and that there is no limit to the number of times it can be populated.
- **Label** — The label for the field as shown in the user interface.
- **Required** — A boolean value showing whether the entry form in the user interface requires a value for the field.
- **Help** — Additional details about working with the field.

Note: You may notice that sometimes the Cardinality value for a field is 0..1, although the Required value is True. The reverse situation can also occur, in which Cardinality is 1..1 but the Required column shows a value of False. In these cases, typically you must meet the most restrictive standard and supply a value for the field.

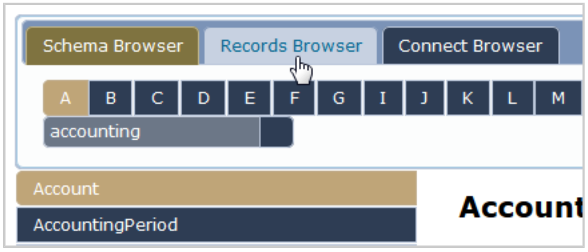
Other Data

At the bottom of each record reference page, you can find the following:

- A list of attributes related to this record. For example, if you want to know whether the record supports External Id, check this area.
- A list of search objects related to this record. For more details about any search object, click the name of the object to view a reference page about it. In response, the page updates to include, in the center pane, details about this search object and, at the left, a tabbed list of other objects from the namespace. For help with this tabbed list, see Navigating to a Different Object.

Comparing SuiteTalk, SuiteScript, and SuiteAnalytics Connect Exposure

To check whether the record you are currently viewing is also supported in SuiteScript or SuiteAnalytics Connect, click the Records Browser tab or the Connect Browser tab at the top of the page.



If the record is supported in SuiteScript or SuiteAnalytics Connect, you are directed to the corresponding page in the SuiteScript Records Browser or SuiteAnalytics Connect Browser. Otherwise, you are directed to the first page of the respective browser.

To compare record types support across SuiteScript, SuiteTalk, and SuiteAnalytics Connect, see the help topic SuiteCloud Supported Records.

Navigating to Other Records

To navigate to a different record, select a different record from the pane at the left. If appropriate, use the A-Z index to refresh the pane so it shows a different list of records.

Navigating to the Default View

If you have navigated to the Schema Browser's namespace view and you want to get back to the default view, click any letter in the A-Z index at the top of the browser window.

Namespace View

In the default view of the SuiteTalk Schema Browser, the pane at the left side of the page lists all the records that correspond with the letter you selected in the A-Z index at the top of the page. Meanwhile, the center pane described details of one record. However, at times you may navigate to a different view, one that is specific to a particular namespace. This view includes:

- In the pane at the left, a tabbed list of all objects in the namespace.
- In the center pane, a summary of details about one object, which can be a record or something else.

This topic describes this view in more detail.

Summary of the Object

In the central pane, the Schema Browser offers a summary of whichever object you have selected in the pane at the left.

In some cases, this object is a record. For help understanding the summary of a record, see Summary of the Record.

If you have selected something other than a record, the exact details shown can vary. For example, the description of an enumeration shows all of its possible values. By contrast, an object such as ExpenseReportExpense, which details a single line in a sublist, is described in a table similar to that of a record.

Navigating to a Different Object

When in the namespace view, you may want to navigate to a different object in the namespace. You can do this using the tabbed list in the pane at the left. The following table describes each subtab.

Tab	Shows
Record	All records and subrecords defined in the namespace
Search	All search objects in the namespace
Other	The namespace's sublists, their child elements, and sometimes other objects
Enum	All enumerations in the namespace

After you click a subtab, the list of buttons at the left updates to reflect the category you chose. Select any button to see the summary of that object.

Navigating to the Namespace View

If you have navigated away from the namespace view and you want to get it back to it, you have the following options:

- Select a namespace using the dropdown list directly below the A-Z index.

- In the center pane, click the namespace link displayed below the name of the record.
- In the **Type** column, click the name of any data type that is underlined.
- At the bottom of the record reference page, click the name of any search object listed.

Entities

The following entity records are supported in SuiteTalk:

- Customer
- Customer Status
- Contact
- Employee
- Group (Entity Group)
- HCM Job
- Partner
- Project (Job)
- Project Type (Job Type)
- Project Status (Job Status)
- Vendor
- Entity Search

Customer

Customer records represent people or companies that purchase goods and services from your business. Use the customer record to manage these customers. Client applications can create, update, or delete customer records. For more details, see the help topic Customers.

The Customer record is defined in the listRel (relationship) XSD.

Supported Operations

The following operations can be used to modify Customer records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's customer reference page. For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Note: The monthlyClosing field is used only by the Japan Edition of NetSuite.

Usage Notes

Usage notes are provide for the following topics:

- Understanding Customer Stages
- Customer Status and Stage Internal IDs
- Returning a Contact List for a Customer
- Working with Customer Sublists

Understanding Customer Stages

In the NetSuite UI, there are three possible stages that can be defined for a customer — lead, prospect or customer. When entering new customers into NetSuite, you can create the record as a lead, prospect or customer record, or you can create the customer as a lead and allow NetSuite to automatically update the stage as certain criteria are met.

- **Lead** — lets you track all the information you need to convert a lead into a customer. Leads have no estimates, opportunities or transactions associated with them. If an estimate or opportunity is created for a lead, the lead becomes a prospect. If you create a sales transaction for a lead, the lead becomes a customer.

> **Note:** When adding leads through web services, you can use the rules you have defined in the SFA (Sales Force Automation) feature to assign leads. Be aware that SFA rules are applied regardless of the setting for the “Use Conditional Defaults on Add” preference.

- **Prospect** —lets you track all the information you need to convert a prospect into a customer. Prospects have no sales orders, invoices, cash sales or other sales transactions associated with them. They can have opportunities and estimates associated with them, however. If a sales transaction is created for a prospect, or an opportunity is closed for a prospect, the prospect becomes a customer.
- **Customer** — lets you track all the information on your current customers.

This workflow is also maintained when working with web services. The Stage field indicates whether the customer is a lead, prospect or customer as defined above. For example, if a customer record is defined as a Lead via web services and then that customer record has an opportunity record associated with it via web services, the customer record Stage field will automatically be updated to reflect the new prospect stage. After the opportunity record is updated to Closed-Won, the record Stage field is automatically updated to Customer. To enter a new customer record at a specified stage, set the Stage field to Lead, Prospect or Customer as desired.

Customer Status and Stage Internal IDs

The following table lists the internal IDs for all standard Customer Status and Stage values that can be used to populate the entityStatus or stage field.

> **Note:** In addition to the following standard custom status values, your organization may have defined *custom* Customer Status values. If the Show Internal IDs preference is enabled, you can confirm the internal ID values in the associated list. For more details, see the help topic Enabling the Show Internal IDs Preference

Customer Statuses		**Customer Stages**	
ID	**Status**	**ID**	**Stage**
17	Dead	LEAD	Lead
6	New	PROSPECT	Prospect
18	Qualified	CUSTOMER	Customer

Customer Statuses		Customer Stages	
ID	**Status**	**ID**	**Stage**
14	Closed Lost		
9	Identified Decision Makers		
8	In Discussion		
11	In Negotiation		
7	Opportunity Identified		
10	Proposal		
12	Purchasing		
13	Closed Won		
16	Lost Customer		
15	Renewal		

Returning a Contact List for a Customer

To return a list of contacts associated to a specific customer record, you must first get the customer record and then perform a search for the contacts associated to that customer.

Sample Java Code

```
// First get your customer
Customer c = (Customer)port.get(new RecordRef("17",RecordType.customer)).getRecord();

// Now do a specific search for the Contacts
ContactSearch cts = new ContactSearch();

// Search for an exact match between Customer.EntityId and the ContactSearch Field Company.cts.
setCompany(new SearchStringField(c.getEntityId(),SearchStringFieldOperator.is));

// Execute the search and you have your contactlist.
SearchResult scts = port.search(cts);
```

Working with Customer Sublists

The SuiteTalk Schema Browser includes all sublists associated with the customer record. See the following information for usage notes regarding specific customer sublists. Usage notes are not provided for every sublist type.

- CustomerContactRoleList
- CustomerCreditCardsList
- CustomerCurrencyList
- CustomerAddressbookList
- CustomerSalesTeamList
- CustomerDownloadList
- SubscriptionsList

For general information about working with sublists in SuiteTalk, see the help topic Sublists in Web Services.

CustomerContactRoleList

This sublist is available for updates of customer records only. You can update this sublist's data to provide Customer Center access to contacts. You can provide access to contacts that already exist in NetSuite and that have already been attached to a customer that already exists in NetSuite. You cannot use this sublist to attach new contacts. The workflow is as follows: 1) Add customer. 2) Add contacts. 3) Attach contacts to customer. 4) Update customer with contact access information with this sublist.

The fields in this sublist map to the fields on the System Information, Access subtab in the UI. These fields include: a Boolean field that indicates whether a contact has access to NetSuite, contact name key field, email address used to log in to NetSuite, password used to log in to NetSuite, and NetSuite role (Customer Center) .

Operations on this sublist do not preserve row order. In some cases, rows added to the sublist will not be returned in the same order.

Note: As of the 2011.2 endpoint, the CustomerContactList sublist is no longer supported. You can use the CustomerContactRoleList instead. In addition, you can add or update contacts through the attach/detach operations, and you can retrieve contact names and roles for an entity through an advanced search on the entity record and its associated contacts. This advanced search technique is described in Returning a Contact List for a Customer.

CustomerCreditCardsList

The creditCardsList field is a list field that lets you provide multiple credit card entries for a customer. The customer can then use these entries for payments. These fields are all mapped to the Financial tab in the UI.

When working with credit card data, be aware of the following security features:

- When adding a credit card, you cannot identify the credit card number using a masked value such as ************5151. You must enter the full 16-digit number, or you can identify an existing credit card record through a RecordRef. (You can identify the full number using the **ccNumber** field. You can reference an existing record using the **creditCard** field.)
- Searches do not work if they include the operator **is** or **isNot** in conjunction with the ccNumber field. The only search operators that can apply to this field are **empty** and **notEmpty**.

CustomerCurrencyList

This list enables you to define multiple currencies that can be used for a customer's transactions, in addition to the primary currency set for a customer. This list is available when the Multiple Currencies feature is enabled. For details about using this feature, see the help topic Multiple Currencies and Customers.

This list's Currency field maps to the Currency field on the Currencies subtab on the Financial subtab of the standard customer record in the UI.

CustomerAddressbookList

The addressBookList field is a list field that lets you provide multiple addresses for a customer. These fields are all mapped to the Address tab in the UI.

When you work with the addressbookList on customer records in the 2009.2 endpoint and beyond, be sure to use **internalId** as the key. Do not use **label**. For the 2009.1 and lower endpoints, the addressbookList is not a keyed sublist, but you can use label to identify records.

Important: Sales territory assignments are based on customers' default billing addresses. If you do not set defaultBilling to True for a customer address, it is not assigned to a territory automatically.

CustomerSalesTeamList

This list is only available when the Team Selling feature is enabled.

CustomerDownloadList

The Sell Downloadable Items feature must be enabled in an account to provide downloads for customers in the Customer Center.

SubscriptionsList

The subscriptionsList sublist on the customer record can be updated. Use the replaceAll flag to update specific lines in this sublist, using **subscription** as the key. Be aware that web services updates to this sublist follow the logic that is permitted in the NetSuite UI. After an entity is unsubscribed from a category, only the entity itself can re-subscribe.

Customer Status

The customer status record describes a lead, prospect, or a customer's stage in the sales cycle. To create a new customer status record, go to Setup > Sales > Setup Tasks > Customer Statuses > New.

The customer status record is defined in the listRel (relationship) XSD.

Supported Operations

The following operations can be used with customer status records:

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's customer status reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Contact

Contacts represent people or companies that you deal with in the daily activity of your business. Use the contact record to create, modify, or delete contacts and associate a contact to a parent record.

The contact record is defined in the listRel (relationship) XSD.

Supported Operations

The following operations can be used to manipulate the contact record.

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's contact reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Working with Contact Sublists

The SuiteTalk Schema Browser includes all sublists (lists) associated with the contact record. See the following information for usage notes regarding specific contact lists. Note that usage notes are not provided for every list type.

AddressbookList

The addressBookList field is a list field that lets you provide multiple addresses for a contact. These fields are all mapped to the Address tab in the UI.

When you work with the addressbookList in the 2009.2 endpoint and beyond, be sure to use **internalId** as the key. Do not use **label**. For the 2009.1 and lower endpoints, the addressbookList is not a keyed sublist, but you can use label to identify records.

Employee

Employee records represent your employees. Use the employee record to store information for contact, login, payroll and human resources purposes.

The employee record is defined in the employees XSD.

Supported Operations

The following operations can be used to modify employee records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's employee reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Entering Expense Reports in Web Services

Web services support for expense reports is different from user interface behavior. In the user interface, expense reports can only be entered for active employees. If an employee's release date is before the current date, expense reports are not supported. In web services, expense reports can be entered for employees that have a release date before the current date.

Specifying Employee Jobs and Positions

When you work with employee jobs and positions, you must submit a value for the workAssignment field to distinguish between a job and a position. By default, the workAssignment field is set to job. If you submit values for the hcmPosition sublist, you must override the value of the workAssignment field, otherwise the sublist values will not be used.

EmployeeAddressbookList

When you work with the addressbookList in the 2009.2 endpoint and beyond, be sure to use **internalId** as the key. Do not use **label**. For the 2009.1 and lower endpoints, the addressbookList is not a keyed sublist, but you can use label to identify records.

Group (Entity Group)

The group record is defined in listRel (relationships) XSD.

Specify the group type (contact, customer, employee, partner, or vendor) by setting the type element on an add operation. Note that the group record is not customizable. Also note that in NetSuite, groups can be dynamic or static. To create a dynamic group, on the group record you must set the group type to dynamic. You can then reference a saved search when adding the group record. The members will change based on the results of the saved search. To create a static group, first add the group record to NetSuite. Then use the **attach** operation to associate members with it. (For information on the attach operation, see the NetSuite Help Center.)

Supported Operations

The following operations can be used with the group record.

add | addList | attach / detach | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's entity group reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

When searching for members of a **dynamic** group, you must use the GroupMemberSearchBasic search interface; you cannot use the group's corresponding entity search. The GroupMemberSearch interface retrieves members of a dynamic group by referencing the groupId.

GroupMemberSearchBasic

Field Name	Type
groupId	platformCore:RecordRef

Field Name	Type
groupId	platformCore:SearchMultiSelectField

The following code snippet shows GroupMemberSearchBasic in a SOAP request.

SOAP Request

```
<soapenv:Body>
   <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <searchRecord xsi:type="ns7:GroupMemberSearchBasic" xmlns:ns7="urn:common_2017_1.platform
.webservices.netsuite.com">
         <ns7:groupId internalId="163" xsi:type="ns8:RecordRef" xmlns:ns8="urn:core_2017_1.plat
form.webservices.netsuite.com"/>
      </searchRecord>
   </search>
</soapenv:Body>
```

If the entity group is not dynamic, you can use the standard EntityGroupSearch interface to execute a search against the group.

HCM Job

Important: This topic describes a feature that is currently available only in some customer accounts. For information on the availability of this feature for your account, please contact your NetSuite account manager.

The HCM Job record is available when the Job Management feature is enabled on the Employees subtab at Setup > Company > Setup Tasks > Enable Features.

In the user interface, you can access the job record at Setup > HR Information System > Position & Job Management > Jobs. For information about working with this record in the user interface, see the help topic Managing Jobs.

The HCM Job record is defined in the employees XSD.

Supported Operations

The following operations can be used with the HCM Job record.

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's HCM Job reference page.

For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

The following request adds a new HCM job record.

SOAP Request

```
<?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>
    <ns1:applicationInfo soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next" soapenv:mus
tUnderstand="0" xmlns:ns1="urn:messages_2017_2.platform.webservices.netsuite.com">
      <ns1:applicationId>APPLICATION ID</ns1:applicationId>
    </ns1:applicationInfo>
    <ns2:passport soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next" soapenv:mustUnders
tand="0" xmlns:ns2="urn:messages_2017_2.platform.webservices.netsuite.com">
      <ns3:email xmlns:ns3="urn:core_2017_2.platform.webservices.netsuite.com">EMAIL</ns3:email
>
      <ns4:password xmlns:ns4="urn:core_2017_2.platform.webservices.netsuite.com">PASSWORD</ns4
:password>
      <ns5:account xmlns:ns5="urn:core_2017_2.platform.webservices.netsuite.com">ACCOUNT</ns5:a
ccount>
      <ns6:role internalId="ROLE" xmlns:ns6="urn:core_2017_2.platform.webservices.netsuite.com"
/>
    </ns2:passport>
  </soapenv:Header>
  <soapenv:Body>
    <add xmlns="urn:messages_2017_2.platform.webservices.netsuite.com">
      <record xsi:type="ns7:HcmJob" xmlns:ns7="urn:employees_2017_2.lists.webservices.netsuite.
com">
        <ns7:title xsi:type="xsd:string">Senior Sales Engineer</ns7:title>
        <ns7:isInactive xsi:type="xsd:boolean">false</ns7:isInactive>
        <ns7:employmentCategory internalId="-10" xsi:type="ns8:RecordRef" xmlns:ns8="urn:core_2
017_2.platform.webservices.netsuite.com"/>
        <ns7:description xsi:type="xsd:string">Sales engineer with 5+ years of experience</ns7:
description>
      </record>
    </add>
  </soapenv:Body>
</soapenv:Envelope>
```

Partner

A partner is a company you have a business agreement with who is not a customer or a vendor. To use partner records the Partner Relationship Management feature must be enabled at Setup > Enable Features > CRM. For more details, see the help topic Creating a Partner Record.

The partner record is defined in the listsRel (relationships) XSD.

Supported Operations

The following operations can be used with partner records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s partner reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Setting a Partner as an Individual or a Company

When creating a partner record, you can specify that partner as either an individual or a company by setting the isPerson field to TRUE or FALSE respectively. The available required and non-required fields vary depending on the setting of this field as detailed in the field definitions table.

Giving Partners NetSuite Access

You can update the PartnerContactRole sublist to provide Partner Center access to contacts. You can give this access to contacts attached to partners. This sublist is available for updates of Partner records only. Partners and contacts must already exist in NetSuite and contacts must already be attached to partners You cannot use this sublist to attach new contacts. The workflow is as follows: 1) Add partner. 2) Add contacts. 3) Attach contacts to partner. 4) Update partner with contact access information from this sublist.

The fields in this sublist map to the fields on the System Information, Access subtab in the UI. These fields include: a Boolean field that indicates whether a contact has access to NetSuite, contact name key field, email address used to log in to NetSuite, password used to log in to NetSuite, and NetSuite role (Partner Center) .

Operations on this sublist do not preserve row order. In some cases, rows added to the sublist will not be returned in the same order.

Note: As of the 2011.2 endpoint, the PartnerContactList sublist is no longer supported. You can use the partnerContactRoleList instead. In addition, you can add or update contacts through the attach/detach operations, and you can retrieve contact names and roles for an entity through an advanced search on the entity record and its associated contacts. This advanced search technique is described in Returning a Contact List for a Customer.

Project (Job)

You use the project record to manage company initiatives.

The project record is defined in the listRel (relationships) XSD, where it is called *Job*.

To use the project record, you must have the Projects feature enabled at Setup > Company > Enable Features, on the Company tab. If you plan to do advanced project tracking, you must also enable Project Management. If you do not see the Project Management check box, your company must first purchase the Project Management add-on from NetSuite.

To access the project record in the UI, choose Lists > Relationships > Projects (or Jobs). For help working with projects manually, see the help topic Projects.

For more on using SuiteTalk to interact with projects, see the following:

- Example 1 — Adding a Project with a Status
- Example 2 — Adding Resources

Important: Starting with the 2008.2 version of NetSuite, the Job record was renamed to Project, but if you have existing code that references the Job record, this code will not break. The name change applies to external UI labels only. You should continue to reference the Job complex type in your code.

Supported Operations

The following operations can be used with the project record:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

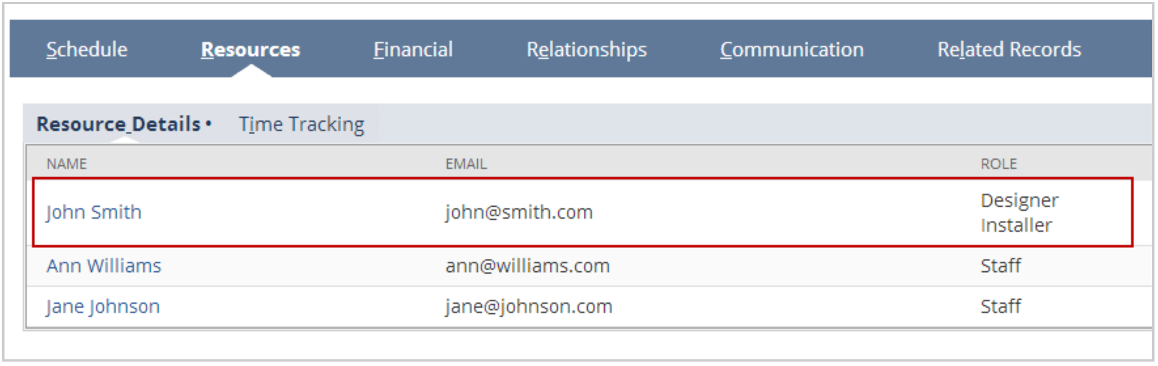
Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's project reference page. For information on using this tool, see SuiteTalk Schema Browser.

Usage Notes

When using SuiteTalk, the behavior of the Resources sublist differs slightly from the behavior in the UI. Specifically, in the UI, each line must have a unique value in the Name field. Further, in the UI, you can select more than one role for each resource.



NAME	EMAIL	ROLE
John Smith	john@smith.com	Designer Installer
Ann Williams	ann@williams.com	Staff
Jane Johnson	jane@johnson.com	Staff

With web services, if you want to add a resource that has three different roles, you set up your code as if you are adding three sublist records for that resource — one for each role. The jobResource values are not required to be unique. To see an example, refer to Example 2 — Adding Resources. Note that after you complete the add operation, the Resources sublist in the UI looks the same as it would if you had manually added one line for the resource, with multiple roles specified on that line, as shown in the illustration above.

Example 1 — Adding a Project with a Status

The following example shows how to add a new project record that has a specific status.

Java

```
public void addJob() throws RemoteException, ExceededRecordCountFault,
        ExceededUsageLimitFault, InsufficientPermissionFault,
        InvalidSessionFault {
                // This operation requires a valid session
                this.login(true);

                Job job = new Job();

                job.setEntityId("XYZ Inc");
                job.setCompanyName("Scripting requirements");

                // Set entity status. The nsKey can be obtained from Setup > Accounting > Accou
nting List
                // Job Status.
                RecordRef status = new RecordRef();
                // 1 = Closed, 2 = In Progress, 3 = Not Awarded, 4 = Pending, 5 = Awarded
                status.setInternalId("4");

                job.setEntityStatus(status);

                // Invoke add() operation
                WriteResponse response = _port.add(job);

                // 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 job was added successfully:"
                        + "\nkey="
                        + ((RecordRef) response.getBaseRef()).getInternalId()
                        + "\nentityId="
                        + job.getEntityId()
                        + "\ncompanyName="
                        + job.getCompanyName()
                        + "\nstatusKey="
                        + job.getEntityStatus().getInternalId());
                } else {
                        _console.error("The job was not added:", true);
                        _console.error(getStatusDetails(response.getStatus()));
                }
        }
```

SOAP Request

```
<soapenv:Body>
      <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
         <record xsi:type="ns6:Job" xmlns:ns6="urn:relationships_2017_1.lists.webservices.netsu
ite.com">
            <ns6:entityId xsi:type="xsd:string">XYZ Inc</ns6:entityId>
            <ns6:companyName xsi:type="xsd:string">Scripting requirements</ns6:companyName>
            <ns6:entityStatus internalId="4" xsi:type="ns7:RecordRef"
              xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com" />
         </record>
      </add>
   </soapenv:Body>
```

SOAP Response

```
 <soapenv:Body>
      <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
         <writeResponse>
            <ns2:status isSuccess="true" xmlns:ns2="urn:core_2017_1.platform.webservices.netsui
te.com" />
            <baseRef internalId="165" type="job" xsi:type="ns3:RecordRef"
              xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com" />
         </writeResponse>
      </addResponse>
   </soapenv:Body>
```

Example 2 — Adding Resources

The following example shows how to create a project that has two resources assigned to it. Note that one resource plays two different roles on the project, so the code is written as if to include a total of three sublist records: two for the person with two roles, plus one for the other resource.

C#

```
private void addProject()
{

   // Create object

   Job newProject = new Job();


   // Set a value for companyName, which represents the name of the
   // project, and for externalId, which is recommended for tracking purposes.

   newProject.companyName = "Launch";
   newProject.externalId = "103A";


```

```
    // If this is a OneWorld account, identify the subsidiary.

    RecordRef subsidiaryRef = new RecordRef();
    subsidiaryRef.internalId = "1";
    newProject.subsidiary = subsidiaryRef;


    // Specify that the Resources sublist will hold three entries.

    newProject.jobResourcesList = new JobResourcesList ();
    newProject.jobResourcesList.jobResources = new JobResources[3];


    // Define the three sublist records. The first two represent the
    // same resource (the resource with the internalId 1148). Two entries
    // are used because this resource is filling two different roles on the project.

    newProject.jobResourcesList.jobResources[0] = new JobResources();
    newProject.jobResourcesList.jobResources[0].jobResource = new RecordRef();
    newProject.jobResourcesList.jobResources[0].jobResource.internalId = "1148";
    newProject.jobResourcesList.jobResources[0].role = new RecordRef();
    newProject.jobResourcesList.jobResources[0].role.internalId = "-3";

    newProject.jobResourcesList.jobResources[1] = new JobResources();
    newProject.jobResourcesList.jobResources[1].jobResource = new RecordRef();
    newProject.jobResourcesList.jobResources[1].jobResource.internalId = "1148";
    newProject.jobResourcesList.jobResources[1].role = new RecordRef();
    newProject.jobResourcesList.jobResources[1].role.internalId = "1";

    newProject.jobResourcesList.jobResources[2] = new JobResources();
    newProject.jobResourcesList.jobResources[2].jobResource = new RecordRef();
    newProject.jobResourcesList.jobResources[2].jobResource.internalId = "1147";
    newProject.jobResourcesList.jobResources[2].role = new RecordRef();
    newProject.jobResourcesList.jobResources[2].role.internalId = "1";

    _service.add(newProject);

}
```

SOAP Request

```
<soap:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.
org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Header>
        <passport xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <email xmlns="urn:core_2017_1.platform.webservices.netsuite.com">user@netsuite.com<
/email>
            <password xmlns="urn:core_2017_1.platform.webservices.netsuite.com">**********</pas
sword>
            <account xmlns="urn:core_2017_1.platform.webservices.netsuite.com">12345</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>
    <soap:Body>
        <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record externalId="103A" xsi:type="q1:Job" xmlns:q1="urn:relationships_2017_1.list
s.webservices.netsuite.com">
                <q1:companyName>Launch</q1:companyName>
                <q1:subsidiary internalId="1"/>
                <q1:jobResourcesList>
                    <q1:jobResources>
                        <q1:jobResource internalId="1148"/>
                        <q1:role internalId="-3"/>
                    </q1:jobResources>
                    <q1:jobResources>
                        <q1:jobResource internalId="1148"/>
                        <q1:role internalId="1"/>
                    </q1:jobResources>
                    <q1:jobResources>
                        <q1:jobResource internalId="1147"/>
                        <q1:role internalId="1"/>
                    </q1:jobResources>
                </q1:jobResourcesList>
            </record>
        </add>
    </soap:Body>
</soap: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_1013519_091820138468207936344769l_943f17897de781</pl
atformMsgs:nsId>
        </platformMsgs:documentInfo>
    </soapenv:Header>
    <soapenv:Body>
        <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <writeResponse>
                <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platf
orm.webservices.netsuite.com"/>
                <baseRef xsi:type="platformCore:RecordRef" type="job" externalId="103A" interna
lId="1155" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </writeResponse>
        </addResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Project Type (Job Type)

Project types are used to define basic project records if the Projects feature is enabled, and the Project Management feature is not enabled. You can use the Projects feature to track information about projects you are working on for customers, including time, contacts, and transactions. In the UI, you can define project types at Setup > Accounting > Accounting Lists > New > Project Type. For more information, see the help topic Basic Projects.

The project type record is defined in listRel (relationships) XSD.

Supported Operations

The following operations can be used with the project type record:

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's project type reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Project Status (Job Status)

The project status record is defined in listRel (relationships) XSD.

Supported Operations

The following operations can be used with the project status record:

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's job status reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Vendor

A vendor is a company or person you purchase goods and services from. Vendor records track information about your vendors and enable you to view past transactions and communications with them.

The vendor record is defined in the listsRel (relationships) XSD.

Supported Operations

The following operations can be used to modify vendor records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's vendor reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Working with Vendor Lists

The SuiteTalk Schema Browser includes all lists (sublists) associated with the vendor record. However, see the following information for usage notes regarding specific vendor lists. Note that usage notes are not provided for every list type.

Note: As of the 2011.2 endpoint, the VendorContactList sublist is no longer supported. You can add or update contacts through the attach/detach operations, and you can retrieve contact names and roles for an entity through an advanced search on the entity record and its associated contacts. This advanced search technique is described in Returning a Contact List for a Customer.

VendorPricingScheduleList

This list can only be edited on an update. It defines the quantity pricing schedules to use for the vendor. The Quantity Pricing feature must be enabled to access this list.

Note about Multiple Currencies and Vendors

When the Multiple Currencies feature is enabled, the label for the Currency field on the vendor record changes to Primary Currency (internal id of the field is still currency) and a Currency sublist is added to the vendor record. Also, the purchasePrice field in ItemVendorList becomes hidden but this field's values can still be saved through web services.

Entity Search

The EntitySearchBasic record is not a stand-alone search record. It must be used in conjunction with a search join from another record. The EntitySearchBasic interface is used when you are unsure of the entity type you are searching against.

EntitySearchBasic is defined in the platformCommon XSD.

Field Definitions

For details, see the Schema Browser’s EntitySearchBasic reference page.

For information on working with search filter fields and search joins, see the search operation in the NetSuite Help Center.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Activities

The following activity records are supported in SuiteTalk:

- Events (CalendarEvent)
- Phone Call
- Project Task
- Resource Allocation
- Tasks

Events (CalendarEvent)

Events are scheduled activities that are automatically added to your calendar when created.

The event record is defined in the actSched (scheduling) XSD.

Supported Operations

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, refer to the Schema Browser’s calendar event reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Events can be scheduled as one-time events that are set to occur on a specific day at a specific time. Events can also be designated as recurring events, which recur over a period of several days, weeks, months, or years. See the following sections for more information:

- Non-Recurring (One-Time) Events
- Recurring Events

Non-Recurring (One-Time) Events

A non-recurring event takes place one time during a single day. Note that a non-recurring event cannot extend beyond midnight in a user's specified time zone. Both the *startDate* and *endDate* fields must contain the same date.

Recurring Events

Recurring events recur on a daily, weekly, monthly, or yearly basis. You can schedule events to automatically recur in your calendar in both basic and advanced patterns.

Important: To add, delete, or update an event with advanced recurrence patterns, you must upgrade to at least the SuiteTalk 2.6 WSDL.

Basic Recurrence Patterns

A basic event recurrence pattern includes events such daily departmental status meetings. These events begin and end on the same day, and they recur in a Monday through Friday pattern. An event with a basic recurrence pattern does not require that you set the recurrenceDow, recurrenceDowim, or recurrenceDowMaskList fields.

To set a basic recurrence pattern:

1. Set the **frequency** field.
2. Set the **period** field.
3. Set the **startDate** field.
4. Set the **endDate** field only if there is a known date by which you want to end the recurring event. Events that recur indefinitely do not require that you set the **endDate** field.

Advanced Recurrence Patterns

Events that are scheduled with advanced recurrence patterns may include some of the following types of patterns:

- A certain number of days apart, such as every 3 days
- The same day of the week with weeks off in between, such as Tuesday every 2 weeks
- The same date of every month
- The same date in a month with months off in between, such as every 3 months on the 20th
- The same day of the week every month, such as the first Friday of every month
- The same day of the week every few months, such as the first Tuesday every other month

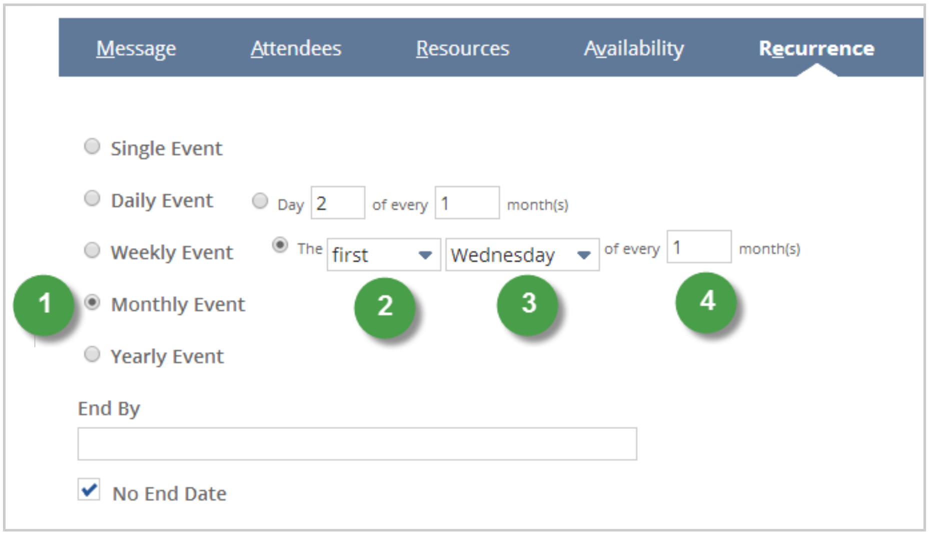
To set an advanced recurrence pattern for an event, you must set the following fields, in addition to setting the *frequency, period*, and *startDate* fields:

- recurrenceDow
- recurrenceDowim
- recurrenceDowMaskList

The recurrenceDow, recurrenceDowim, and recurrenceDowMaskList fields do not have corresponding field labels in the UI. These fields represent a collection of functions which are shown in the figure below.

Note: The recurrenceDow, recurrenceDowim, and recurrenceDowMaskList fields are hidden fields in SuiteScript. These fields are currently supported in server-side SuiteScript development; they are not supported in client development.

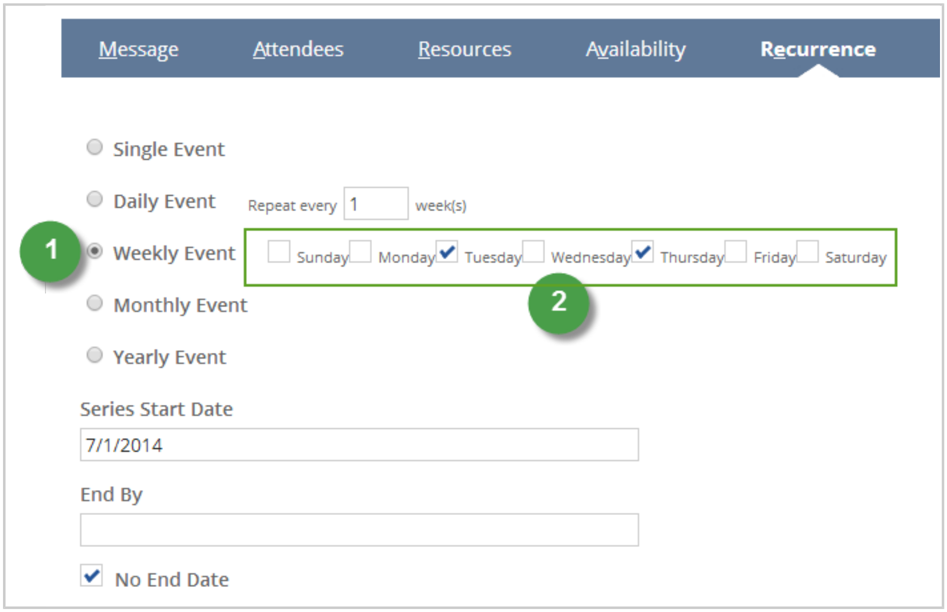
The following screenshot represents the scheduling of a monthly event that contains an advanced recurrence pattern.



The preceding screenshot shows the equivalent of the following settings.

Callout	Description
1	The *frequency* field is set to _**month**.
2	The *recurrenceDowim* field is set to _**first**.
3	The *recurrenceDow* field set to _**wednesday**.
4	The *period* field is set to **1**.

The next screenshot represents the scheduling of a weekly event that contains an advanced recurrence pattern.



The preceding screenshot shows the equivalent of the following settings.

Callout	Description
1	The *frequency* field is set to **_week**.
2	The *recurrenceDowMaskList* field is set to **_tuesday** and **_thursday**.

To set an advanced recurrence pattern that recurs monthly or yearly:

1. Set the **frequency** field to indicate how often the event will recur. The value specified for **frequency** dictates the values that will be set for the **recurrenceDow** and **recurrenceDowim** fields.
2. Next, set the **period** field.
3. Set the value for **recurrenceDow**.
4. Set the value for **recurrenceDowim**.

To set an advanced recurrence patterns that recurs weekly:

1. Set the **frequency** field to indicate how often the event will occur. The value specified for **frequency** dictates the value that will be set for the **recurrenceDow** MaskList.
2. Next, set the **period** field.
3. Finally, set **recurrenceDowMaskList**.

Updating a Series of Events that Include Advanced Recurrence Patterns

When updating a record in web services, generally you submit only the values you intend to change. However, when updating an event record that includes advanced recurrence patterns, you must use nullFieldList to explicitly remove the values that were previously set.

For example, to update a **monthly** or a **yearly** event from a specific day of the week in a month to a specific day of the month, use nullFieldList to explicitly remove the values in recurrenceDowim and recurrenceDow.

Updating a Single Instance of a Recurring Event

A recurring event can be modified to change or delete an individual event in the series. In SuiteTalk this is handled using the **ExclusionDateList** type.

Deleting individual events in a series

To delete individual events in a series, update the recurring event with a list of exclusions (in the form of dates) in the ExclusionDateList type. The ExclusionDateList type sets a list of timestamps of individual events that have been excluded from the series.

Important: After an event has been excluded, it is separated from the rest of the events in the series, and it can NOT be associated with the event series.

To update an individual instance of a recurring event

1. Update the recurring event with a list of exclusions (in the form of dates) in the **exclusionDateList** field.

2. For each exclusion, add a new event that contains the updated data.

   The original event and the new modified event have different internal ID values and the new updated event is no longer associated with the event series.

Retrieving excluded events from a series

To retrieve excluded events from a series, submit a get operation for the desired event series and return the values listed in the ExclusionDateList type.

Working with Event Lists

The SuiteTalk Schema Browser includes all lists (sublists) associated with the Event record. See the following information for usage notes regarding specific Event lists. Note that usage notes are not provided for every list type.

- AttendeeList
- ExclusionDateList
- ResourceList
- TimeItemList

AttendeeList

This list is used to identify each attendee for the event.

ExclusionDateList

This list is used to denote individual events in a series that have been modified or deleted. For detailed information, see Updating a Single Instance of a Recurring Event.

ResourceList

This list is used to schedule available resources for the event.

TimeItemList

This list, available when the Time Tracking feature is enabled, is used to track employee time associated with the event, including payroll, billing, and project fields.

Phone Call

Phone calls records are used to document phone call activity. All information submitted for a phone call record is stored on a record in the phone call list, on the customer record who calls and on any contact's records referenced in the call contact list.

The phone call record is defined in the actSched (scheduling) XSD.

When the Time Tracking feature is enabled, the TimeItemList sublist is available. This list is used to track employee time associated with the phone call, including payroll, billing, and project fields.

Supported Operations

The following operations can be used to manipulate the phone call record.

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's phone call reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Project Task

The project task record can be used to keep track of specific activities and milestones associated with a project.

This record is defined in the actSched (scheduling) XSD, where it is called *ProjectTask*.

The project task record is available when the Project Management feature is enabled at Setup > Company > Enable Features, on the Company subtab. When the feature is enabled, you can access the project task record in the UI by navigating to an existing project and clicking the New Project Task or New Milestone button. For details on working with this record in the UI, see the help topic Project Tasks.

Project task records cannot be created as standalone records. Rather, you create a project task for a specific project record, and the task remains attached to that record. For information on working with the project record in web services, see Project (Job).

For more on using web services to interact with the project task record, see the following:

- Project Tasks Versus Milestone Tasks
- Project Task Supported Operations
- Project Task Field Definitions
- Project Task Permissions
- Example — Adding a Project Task
- Example — Adding a Milestone Task

Project Tasks Versus Milestone Tasks

Every project task record is designated as either a project task or a milestone task. A project task is used to represent an activity, whereas a milestone task is used to represent a checkpoint in the overall progress of the project.

Although they are the same type of record, a milestone task cannot have values in the estimatedWork body or sublist fields. Therefore, if you create a project task record and do not include any estimated work, the record is automatically saved as a milestone task. If you do include estimated work, the record is saved as a project task.

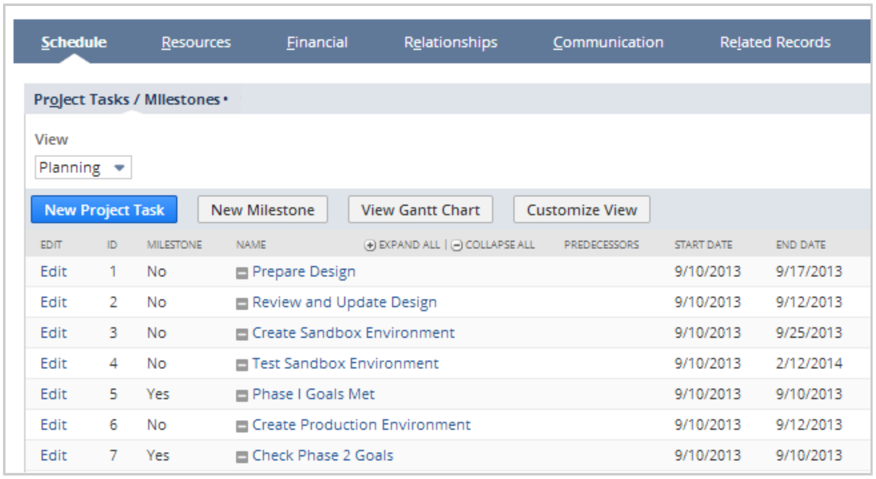
These same rules apply to the updating of records as well. In other words:

- To convert a project task into a milestone task, clear the estimatedWork body field and all values from the Assignees sublist.
- To convert a milestone task into a project task, add a value to the estimatedWork body field or add at least one record to the Assignees sublist, with a positive value of estimated work.

Note that the estimatedWork body field is populated by the sum of estimated work listed for Assignees. If you explicitly set a value for the estimatedWork body field, and you also include Assignees, the value you specify for the body field is overwritten based on the sublist's estimatedWork values. If you do not include Assignees, you can explicitly assign a value to the estimatedWork body field.

Another difference is that a milestone task cannot have a value in the finishByDate field, which in the UI is labeled Finish No Later Than. Therefore, if the system determines that your task is a milestone task and you try to set a value for Finish No Later Than, the system discards the value.

On the project record, project tasks and milestone tasks are listed together on the Schedule tab. You can tell which tasks are milestones by referring to the Milestone column.



EDIT	ID	MILESTONE	NAME	PREDECESSORS	START DATE	END DATE
Edit	1	No	Prepare Design		9/10/2013	9/17/2013
Edit	2	No	Review and Update Design		9/10/2013	9/12/2013
Edit	3	No	Create Sandbox Environment		9/10/2013	9/25/2013
Edit	4	No	Test Sandbox Environment		9/10/2013	2/12/2014
Edit	5	Yes	Phase I Goals Met		9/10/2013	9/10/2013
Edit	6	No	Create Production Environment		9/10/2013	9/12/2013
Edit	7	Yes	Check Phase 2 Goals		9/10/2013	9/10/2013

Project Task Supported Operations

The following operations are supported for use with the project task record:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Regarding the attach operation, note that you can attach files, but not contact records.

Project Task Field Definitions

This topic describes some of the important body fields and sublist fields available for the project task record.

Body Fields

This section lists some of the key body fields available for this record.

company

The company field represents the project that this task is associated with. Every task must be attached to a particular project, so this field is required.

estimatedWork

The estimatedWork body field is populated in one of two ways:

- You can explicitly specify a value for this field — but you should use this approach only if you do not plan add values to the Assignees sublist, or if you are not yet ready to populate the sublist.
- You can enter values for the Assignees sublist, and the system uses a value derived from the sublist to populate the estimatedWork body field. In this case, the system discards any value you might have specified for the estimatedWork body field. The reason is that each Assignee in the sublist has an estimatedWork value. So if you enter values in this sublist, the system totals the Estimated Work values for each assignee and uses this sum as the value for the Estimated Work body field.

Whether or not you include a value for this field determines whether this record is saved as a project task or a milestone task. For more details on this distinction, see Project Tasks Versus Milestone Tasks.

startDate

You cannot set a value for startDate unless you have also set the constraintType body field to _fixedDate. Refer also to Example — Adding a Project Task.

title

The title field is the name of the Task. In the UI, this field is labeled Name. The value you choose for this field is the one used to represent the project task on the project record. This field is required.

Sublist Fields

Two sublists are available for integration — Time Tracking and Assignees.

Assignees

You can use the Assignees sublist (assigneeList) to assign employees or vendors to a project task. Note that the only resources available for adding to the sublist are those listed as resources on the project record.

When you add a record to the Assignees sublist, each record must include values for the following:

- resource — Used to identify the assignee.
- estimatedWork — The total hours the resource is expect to invest. The system uses the sum of all the estimatedWork sublist values to populate the estimatedWork body field.

- unitCost — The cost per hour of the resource's time.

If you are adding a record and you omit one of these values for any line in the sublist, the operation fails.

Note: In order for an employee or vendor to be added as a resource to a project record, they must have been flagged as a project resource on the employee or vendor record. For more details, see the help topic Working with Resources in Project Management.

Time Tracking

When the Time Tracking feature is enabled, the Time Tracking sublist (timeItemList) is available. This list is used to track employee hours associated with the task.

For More Information

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's project task reference page. For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Project Task Permissions

Even if the Advanced Project feature is enabled, users must have the Edit or Full permission for project tasks before they can create, update, or delete this record. For search, the user must have at least the View permission. NetSuite administrators can enable the **Project Task** permission for specific roles by going to Setup > Users/Roles > Manage Roles.

After you open a role for editing, note that the project task record is listed on the Lists subtab of the Permissions tab.

Example — Adding a Project Task

The following example shows how to add a project task record with one record in the Assignees sublist.

C#

```
private void addProjectTask()
{


   //Create object

   ProjectTask newProjectTask = new ProjectTask();


   // Set a value for title, which is required, and externalId, which is recommended.

   newProjectTask.title = "Migrate Data";
   newProjectTask.externalId = "125W";
```

```
    //Optionally, set a value for the startDate field.

    newProjectTask.constraintType = ProjectTaskConstraintType._fixedStart;
    newProjectTask.constraintTypeSpecified = true;
    DateTime nextWeek = new DateTime(2013, 9, 30);
    newProjectTask.startDate = nextWeek;
    newProjectTask.startDateSpecified = true;


    // Optionally, set a value for the finishByDate field.

    DateTime nextMonth = new DateTime(2013, 11, 11);
    newProjectTask.finishByDate = nextMonth;
    newProjectTask.finishByDateSpecified = true;


    // Create values in the Assignees sublist.

    newProjectTask.assigneeList = new ProjectTaskAssigneeList();
    newProjectTask.assigneeList.projectTaskAssignee = new ProjectTaskAssignee[2];

    newProjectTask.assigneeList.projectTaskAssignee[0] = new ProjectTaskAssignee();
    newProjectTask.assigneeList.projectTaskAssignee[0].resource = new RecordRef();
    newProjectTask.assigneeList.projectTaskAssignee[0].resource.internalId = "1149";
    newProjectTask.assigneeList.projectTaskAssignee[0].estimatedWork = 30;
    newProjectTask.assigneeList.projectTaskAssignee[0].estimatedWorkSpecified = true;
    newProjectTask.assigneeList.projectTaskAssignee[0].unitCost = 60;
    newProjectTask.assigneeList.projectTaskAssignee[0].unitCostSpecified = true;

    newProjectTask.assigneeList.projectTaskAssignee[1] = new ProjectTaskAssignee();
    newProjectTask.assigneeList.projectTaskAssignee[1].resource = new RecordRef();
    newProjectTask.assigneeList.projectTaskAssignee[1].resource.internalId = "1147";
    newProjectTask.assigneeList.projectTaskAssignee[1].estimatedWork = 15;
    newProjectTask.assigneeList.projectTaskAssignee[1].estimatedWorkSpecified = true;
    newProjectTask.assigneeList.projectTaskAssignee[1].unitCost = 75;
    newProjectTask.assigneeList.projectTaskAssignee[1].unitCostSpecified = true;


    // Every project task must be associated with a project record. Identify that record here.

    RecordRef projectRef = new RecordRef();
    projectRef.internalId = "1146";
    newProjectTask.company = projectRef;

    _service.add(newProjectTask);

}
```

SOAP Request

```
<soap:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.
```

```
org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Header>
        <passport xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <email xmlns="urn:core_2017_1.platform.webservices.netsuite.com">user@netsuite.com<
/email>
            <password xmlns="urn:core_2017_1.platform.webservices.netsuite.com">**********</pas
sword>
            <account xmlns="urn:core_2017_1.platform.webservices.netsuite.com">12345</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>
    <soap:Body>
        <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record externalId="125W" xsi:type="q1:ProjectTask" xmlns:q1="urn:scheduling_2017_1
.activities.webservices.netsuite.com">
                <q1:title>Migrate Data</q1:title>
                <q1:company internalId="1146"/>
                <q1:constraintType>_fixedStart</q1:constraintType>
                <q1:startDate>2013-09-30T00:00:00</q1:startDate>
                <q1:finishByDate>2013-11-11T00:00:00</q1:finishByDate>
                <q1:assigneeList>
                    <q1:projectTaskAssignee>
                        <q1:resource internalId="1149"/>
                        <q1:estimatedWork>30</q1:estimatedWork>
                        <q1:unitCost>60</q1:unitCost>
                    </q1:projectTaskAssignee>
                    <q1:projectTaskAssignee>
                        <q1:resource internalId="1147"/>
                        <q1:estimatedWork>15</q1:estimatedWork>
                        <q1:unitCost>75</q1:unitCost>
                    </q1:projectTaskAssignee>
                </q1:assigneeList>
            </record>
        </add>
    </soap:Body>
</soap: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_1013519_0917201373149825972664381_be111cd446b940</pl
atformMsgs:nsId>
        </platformMsgs:documentInfo>
    </soapenv:Header>
    <soapenv:Body>
        <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <writeResponse>
                <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platf
orm.webservices.netsuite.com"/>
```

```
                <baseRef xsi:type="platformCore:RecordRef" type="projectTask" externalId="125W"
 internalId="3218" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </writeResponse>
        </addResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Example — Adding a Milestone Task

The following example shows how to add a milestone task.

C#

```
private void addMilestone()
{


   //Create object

   ProjectTask newMilestoneTask = new ProjectTask();


   // Set a value for title, which is required, and externalId, which is recommended.
   // Because this is a milestone task, omit an estimatedWork value.

   newMilestoneTask.title = "Phase I Goals Met";
   newMilestoneTask.externalId = "125I";


   // Every milestone task must be associated with a project record. Identify that record here.


   RecordRef projectRef = new RecordRef();
   projectRef.internalId = "1146";
   newMilestoneTask.company = projectRef;

   _service.add(newMilestoneTask);

}
```

SOAP Request

```
<soap:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.
org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Header>
        <passport xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <email xmlns="urn:core_2017_1.platform.webservices.netsuite.com">user@netsuite.com<
/email>
            <password xmlns="urn:core_2017_1.platform.webservices.netsuite.com">**********</pas
sword>
            <account xmlns="urn:core_2017_1.platform.webservices.netsuite.com">12345</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>
    <soap:Body>
        <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record externalId="125I" xsi:type="q1:ProjectTask" xmlns:q1="urn:scheduling_2017_1
.activities.webservices.netsuite.com">
                <q1:title>Phase I Goals Met</q1:title>
                <q1:company internalId="1146"/>
            </record>
        </add>
    </soap:Body>
</soap: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_1013519_091320133994781011268665637_780d0420bb560</pl
atformMsgs:nsId>
        </platformMsgs:documentInfo>
    </soapenv:Header>
    <soapenv:Body>
        <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <writeResponse>
                <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platf
orm.webservices.netsuite.com"/>
                <baseRef xsi:type="platformCore:RecordRef" type="projectTask" externalId="125I"
 internalId="2821" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </writeResponse>
        </addResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Project Task Assignment Joined Search

The ProjectTaskAssignmentSearchBasic record is not a stand-alone search record. It must be used in conjunction with a search join from the projectTaskSearch record.

The ProjectTaskAssignmentSearchBasic is defined in the platformCommon XSD.

Field Definitions

See SuiteTalk Schema Browser for a list of all project task assignment search filter fields.

For information on working with search filter fields and search joins, see the search () operation in the NetSuite Help Center.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Resource Allocation

Important: For information on the availability of the Resource Allocations feature, please contact your account representative.

The resource allocation record lets you reserve an employee's time for a particular project.

The resource allocation record is defined in the actSched (scheduling) XSD, where it is called *resourceAllocation*.

ResourceAllocationSearch, ResourceAllocationSearchBasic, and ResourceAllocationSearchAdvanced are also exposed to SuiteTalk.

In the UI, you access the resource allocation record by going to Activities > Scheduling > Resource Allocations. Alternatively, you can view the resource allocations for a particular project through the project record's Resources subtab (Lists > Relationships > Projects, or Lists > Relationships > Jobs).

This record is available only if the Resource Allocations feature has been enabled at Setup > Company > Enable Features, on the Company tab. Note that this option will not be visible unless your account has been provisioned for this feature. For more details, contact your account representative.

Supported Operations

The following operations can be used with the resource allocation record.

add | addList | delete | deleteList | get | getDeleted | getList | getSelectValue | search | searchMore | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

To create a new resource allocation record, you must reference two existing NetSuite records, as follows:

- project — A reference to a project record defined at Lists > Relationships > Projects. Note that in some NetSuite accounts, projects are referred to as jobs.
- allocationResource — A reference to an employee record, defined at Lists > Employees > Employees, for which the Project Record option has been checked. The Project Resource box is located on the Human Resources tab of the employee record.

You are also required to provide values for several other fields. Refer to the SuiteTalk Schema Browser for details on all available body fields, sublist fields, search filters, and search joins. For details, see the Schema Browser’s resource allocation reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Code Samples

Refer to the following examples for help interacting with the resource allocation record.

Creating a Record

The following example shows how to create a resource allocation record.

Java

```
public void testAddResourceAllocation() throws Exception
{
    // This operation requires a valid session
    this.login(true);

    ResourceAllocation resourceAllocation = new ResourceAllocation();
    String extId = "TKO_"+rs();
    resourceAllocation.setExternalId(extId);
    RecordRef allocationResourceRef = new RecordRef();
    allocationResourceRef.setInternalId("46");
    resourceAllocation.setAllocationResource(allocationResourceRef);
    RecordRef projectRef = new RecordRef();
    projectRef.setInternalId("43");
    resourceAllocation.setProject(projectRef);
    resourceAllocation.setStartDate(Calendar.getInstance());
    resourceAllocation.setEndDate(Calendar.getInstance());
    resourceAllocation.setAllocationAmount(4.0);
    resourceAllocation.setAllocationUnit(ResourceAllocationAllocationUnit._hours);
    RecordRef allocationTypeHardRef = new RecordRef();
    allocationTypeHardRef.setInternalId("1");
    resourceAllocation.setAllocationType(allocationTypeHardRef);
    c.addRecord(resourceAllocation);
}
```

SOAP Request

```
<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>
     <ns1:passport soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next" soapenv:mustUnder
stand="0" xmlns:ns1="urn:messages_2017_1.platform.webservices.netsuite.com">
        <ns2:email xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">nlbuild@netsui
te.com</ns2:email>
        <ns3:password xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">********</n
s3:password>
        <ns4:account xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com">3604360</ns4
:account>
        <ns5:role internalId="3" xmlns:ns5="urn:core_2017_1.platform.webservices.netsuite.com"/
>
```

```
      </ns1:passport>
   </soapenv:Header>
   <soapenv:Body>
      <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
         <record externalId="TKO_drPIik" xsi:type="ns6:ResourceAllocation" xmlns:ns6="urn:schedu
ling_2017_1.activities.webservices.netsuite.com">
            <ns6:allocationResource internalId="46" xsi:type="ns7:RecordRef" xmlns:ns7="urn:core
_2017_1.platform.webservices.netsuite.com"/>
            <ns6:project internalId="43" xsi:type="ns8:RecordRef" xmlns:ns8="urn:core_2017_1.pla
tform.webservices.netsuite.com"/>
            <ns6:startDate xsi:type="xsd:dateTime">2013-04-08T08:48:30.249Z</ns6:startDate>
            <ns6:endDate xsi:type="xsd:dateTime">2013-04-08T08:48:30.249Z</ns6:endDate>
            <ns6:allocationAmount xsi:type="xsd:double">8.0</ns6:allocationAmount>
            <ns6:allocationUnit xsi:type="ns9:ResourceAllocationAllocationUnit" xmlns:ns9="urn:t
ypes.scheduling_2017_1.activities.webservices.netsuite.com">_hours</ns6:allocationUnit>
            <ns6:allocationType internalId="1" xsi:type="ns10:RecordRef" xmlns:ns10="urn:core_20
17_1.platform.webservices.netsuite.com"/>
         </record>
      </add>
   </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```
<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.ne
tsuite.com">
         <platformMsgs:nsId>WEBSERVICES_3604360_040820133886900932011092328_d580194f6302</platfo
rmMsgs:nsId>
      </platformMsgs:documentInfo>
   </soapenv:Header>
   <soapenv:Body>
      <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
         <writeResponse>
            <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.w
ebservices.netsuite.com"/>
            <baseRef internalId="7" externalId="TKO_drPIik" type="resourceAllocation" xsi:type="
platformCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/
>
         </writeResponse>
      </addResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Updating a Record

The following example shows how to update a resource allocation record.

Java

```
public void testUpdate() throws Exception
```

```
{
    c.useRequestLevelCredentials();
    ResourceAllocation resourceAllocation = new ResourceAllocation();
    resourceAllocation.setInternalId("7");
    resourceAllocation.setNotes("WS");
    c.updateRecord(resourceAllocation);
}
```

SOAP Request

```
<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>
     <ns1:passport soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next" soapenv:mustUnder
stand="0" xmlns:ns1="urn:messages_2017_1.platform.webservices.netsuite.com">
        <ns2:email xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">nlbuild@netsui
te.com</ns2:email>
        <ns3:password xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">********</n
s3:password>
        <ns4:account xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com">3604360</ns4
:account>
        <ns5:role internalId="3" xmlns:ns5="urn:core_2017_1.platform.webservices.netsuite.com"/
>
     </ns1:passport>
  </soapenv:Header>
  <soapenv:Body>
     <update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
        <record internalId="7" xsi:type="ns6:ResourceAllocation" xmlns:ns6="urn:scheduling_2017
_1.activities.webservices.netsuite.com">
           <ns6:notes xsi:type="xsd:string">WS</ns6:notes>
        </record>
     </update>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```
<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.ne
tsuite.com">
        <platformMsgs:nsId>WEBSERVICES_3604360_040520136333772305222772O3_1e0d36f5119c6</platfo
rmMsgs:nsId>
     </platformMsgs:documentInfo>
  </soapenv:Header>
  <soapenv:Body>
     <updateResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
        <writeResponse>
           <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.w
ebservices.netsuite.com"/>
           <baseRef internalId="7" type="resourceAllocation" xsi:type="platformCore:RecordRef"
xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
```

```
            </writeResponse>
        </updateResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Tasks

Tasks are activities that need to be completed. Use the task record to add new tasks for individuals, companies or contacts and to modify those records.

The task record is defined in the actSched (scheduling) XSD.

Supported Operations

The following operations can be used with task records.

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s task reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

In web services tasks are ordered by creation date. In the UI, tasks can be ordered by specifying an Insert Before parameter that places the current task before an existing task assigned to the same user. This functionality is NOT available through web services.

Identifying Unique TaskContact List Records

Entries in the TaskContact list are identified by unique combinations of the customer/contact pairing. Therefore, if you have multiple entries in the TaskContact list with the same customer, the contact is also needed to further identify the record. In the event that two or more entries have the same customer/contact pairing, these entries are treated as a single unit.

Time Tracking Sublist

When the Time Tracking feature is enabled, the TimeItemList sublist is available. This list is used to track employee time associated with the task, including payroll, billing, and project fields.

Restricting Time Entry on Tasks

Unless the web services user is an administrator or the employee specified on the task, the user will not be able to set the reminderType and reminderMinutes fields on a task record. If the user attempts to set these fields, the values specified will not be saved.

In the UI, you can set the **Check the Limit Time to Assignees** box to restrict time entry against this job. When you check this box, only assigned resources can enter time for this job's tasks.

Setting Start and End Times for Timed Tasks

In web services the startTime and endTime field are not exposed. Therefore, to set start and end times for a timed task, users must first set the timedEvent field to TRUE. Then the time component of the DateTime value that is specified in startDate will be set for startTime, and the time component of the DateTime value that is specified in endDate will be set for endTime.

Note that the fields reminderMinutes and reminderType are also only settable if timedEvent is set to TRUE.

Contacts Sublist

TaskContactList, which lists contacts and companies related to each task, does not have a unique key field, so it can include duplicated values.

If you set the ReplaceAll preference to True for a write to the sublist, and a value that you submit matches two nonunique existing values, an error is thrown, because it is not clear which existing value to match. If a value you submit matches a unique row, the row is preserved and no error is thrown.

Code Sample

The following sample shows how to add a task to a job record.

Java

```
public void addTaskToJob() throws RemoteException {
                this.login(true);

                RecordRef custRef = new RecordRef();
                custRef.setInternalId("95");
                custRef.setType(RecordType.job);

                Task task = new Task();
                task.setCompany(custRef);
                task.setTitle("Race Car Tuning");
                task.setMessage("Includes baseline dyno, 5 hours tuning session & turbo upgrade
.");

                WriteResponse response = _port.add(task);

                if (response.getStatus().isIsSuccess()) {
                        _console.info("\nThe following task was added successfully: " + ((Recor
dRef) response.getBaseRef()).getInternalId());
                } else {
                        _console.error("The task was not added:", true);
                        _console.error(getStatusDetails(response.getStatus()));
                }
        }
```

Communications

The following communications records are supported in SuiteTalk:

- Note
- Message

Note

Notes are used to attach information to another record. Use the notes record to create new notes and attach them to a specific record.

The note record is defined in the generalComm (communication) XSD.

Supported Operations

The following operations can be used to modify the note record.

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's note reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

A note must be associated with one and only one of these records. If values for more than one of these fields is submitted, an error is thrown.

Sample Code

The following SOAP and C# samples show how to add a note.

SOAP Request

```
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <soap:Header>
      <preferences xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
         <warningAsError>false</warningAsError>
         <useConditionalDefaultsOnAdd>false</useConditionalDefaultsOnAdd>
      </preferences>
   </soap:Header>
   <soap:Body>
      <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
         <record xsi:type="q1:Note" xmlns:q1="urn:communication_2017_1.general.webservices.nets
uite.com">
            <q1:title>-- note title goes here --</q1:title>
            <q1:noteType internalId="7" type="noteType" />
            <q1:direction>_outgoing</q1:direction>
            <q1:noteDate>2008-04-09T00:00:00</q1:noteDate>
            <q1:note>-- memo goes here --</q1:note>
            <q1:activity internalId="39" type="calendarEvent" />
            <q1:author internalId="-5" type="employee" />
         </record>
      </add>
   </soap:Body>
</soap:Envelope>
```

SOAP Response

```
<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:documentInfo xmlns:ns1="urn:messages_2017_1.platform.webservices.netsuite.com">
         <ns1:nsId>WEBSERVICES_721410_040920088803062742117685935_252c2fccbf8a0</ns1:nsId>
      </ns1:documentInfo>
   </soapenv:Header>
   <soapenv:Body>
      <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
         <writeResponse>
            <ns2:status isSuccess="true" xmlns:ns2="urn:core_2017_1.platform.webservices.netsui
te.com" />
            <baseRef internalId="27" type="note" xsi:type="ns3:RecordRef"
              xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com" />
         </writeResponse>
      </addResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

C#

```
private void addNotes()
        {
            this.login(true);

            Note note = new Note();
            RecordRef activityref = new RecordRef();
            activityref.internalId = "39";
            activityref.type = RecordType.calendarEvent;
            activityref.typeSpecified = true;
            note.activity = activityref;

            RecordRef authorref = new RecordRef();
            authorref.internalId = "-5";
            authorref.type = RecordType.employee;
            authorref.typeSpecified = true;
            note.author = authorref;

            note.direction = NoteDirection._outgoing;
            note.directionSpecified = true;

            RecordRef notetyperef = new RecordRef();
            notetyperef.internalId = "7";
            notetyperef.type = RecordType.noteType;
            notetyperef.typeSpecified = true;
            note.noteType = notetyperef;

            DateTime searchDate = new DateTime();
            searchDate = DateTime.Now;
            searchDate = DateTime.Parse(searchDate.ToString("dd/MM/yyyy"));
            note.noteDate = searchDate;
            note.noteDateSpecified = true;

            note.title = "-- note title goes here --";
            note.note = "-- memo goes here --";

            WriteResponse writeRes = _service.add(note);
            if (writeRes.status.isSuccess)
            {
                _out.writeLn("\nThe note " + note.internalId + " has been added successfully");

            }
            else
            {
                _out.error(getStatusDetails(writeRes.status));
            }
        }
```

Message

A message is used to record correspondence you have with a specific business. Use the message record to add an email message to an existing customer, contact, or opportunity record. After an email message has been added to a record, any related emails are automatically attached to the same record as well as to any recipients of the original email.

The message record is defined in the generalComm (communication) XSD.

For information about adding an email message to a record in the NetSuite user interface, see the help topic Sending Email from Records.

Note: Adding letters, PDFs, or faxes through web services is currently not supported.

Supported Operations

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext |

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's message reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Message RecordRef Fields

Message recordref fields cannot be referred to by externalId.

Email Fields May Be Required

The following Email fields are required if the From or Attach to entity records do not have an email address specified. If you attempt to create a message without both of these fields specified, the request fails with a message about missing mandatory fields.

Field	Label in UI	Description
`authorEmail`	(From) Email Address	A string that includes the email address of the person writing the email. This email address is displayed as the From address in the email message that is sent to the recipient.
`recipientEmail`	(To) Email Address	A string that includes the email address of the person receiving the email. This email address is the address to which the email message is sent..

Using the BCC and CC Fields on the Message Record

The bcc and cc fields represent the list of secondary email addresses associated with the message. These are string fields that accept a space deliminated list of email addresses. For endpoints prior to

2009.2, invalid email addresses in the bcc and cc fields do not result in errors. For the 2009.2 and later endpoints, invalid email addresses in the bcc and cc fields return Invalid Field Value errors.

Using the dateTime Field on the Message Record

The dateTime field is only included in 2009.2 and earlier endpoints. It contains the same value as the messageDate field, but in a different date format. If you use an endpoint later than 2009.2, use messageDate rather than the obsolete dateTime field.

Working with Attachments

You cannot update an attachment in the context of a message, but you can do so through an update operation on the file record. File records represent files that are stored in the NetSuite file cabinet. Use the file record to define email attachments sent or received via the messages record. For additional usage notes on working with the file record, see File.

The file record is defined in the docFileCab (fileCabinet) XSD.

Encoding

The schema has a data type of xsd:base64Binary which is mapped to a byte array in .NET and Axis (Java). On the client side we only need to pass a text or binary file as a byte array with no further base64 encoding. Therefore, do NOT Base64 Encode the file before sending it to Axis or .NET as these layers will do so themselves.

For example:

- Correct

```
--------
fis = new FileInputStream(inFile);
b = new byte[(int)inFile.length()];
fis.read(b);
myMediaItem.setContent(b);
--------
```

- Incorrect

```
---------
fis = new FileInputStream(inFile);
b = new byte[(int)inFile.length()];
fis.read(b);
myMediaItem.setContent(new String(Base64.encode(b)));
```

Storing Attachments

To maintain uniqueness for each file attachment, when attachments are stored in NetSuite two levels of sub-folders are automatically created for the Attachments Sent and Attachments Received respectively.

If the incoming field of the message record is set to **false**, the attachment is saved in the Attachments Sent folder. If the incoming filed is set to **true**, the attachment is saved in the Attachments Received folder.

The folder structure being generated is as follows:

- File Cabinet
- Attachments Sent or Attachments Received
- Entity Name
- Date & Time / Msg ID
- File name

For example:

A user is using auto email reply capture feature. They send a message to their contact John Smith from within the application. When the contact replies to the message they also attach a file to the message. When the message is created in NetSuite (through the auto reply capture feature) it is saved to a sub-folder that is created when the attachment is saved. The path to the sub-folder is as follows:

file cabinet > attachments received > Attachments Received > John Smith > 20050620_Message_525 > File.doc.

Note: If an attachment is being stored with an extension that does not match the actual file type (for example a .txt file as a .exe file), the file is listed in NetSuite as other text or other binary but this is a label — there is no logical effect on the file.

Adding Messages

The following is a basic sample that shows how to add a message record.

SOAP Request

```
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Header>
<preferences xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<warningAsError>false</warningAsError>
<useConditionalDefaultsOnAdd>false</useConditionalDefaultsOnAdd>
</preferences>
</soap:Header>
<soap:Body>
<add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<record xsi:type="q1:Message" xmlns:q1="urn:communication_2017_1.general.webservices.netsuite.c
om">
<q1:author internalId="-5" type="employee" />
<q1:recipient internalId="-5" type="employee" />
<q1:messageDate>2008-04-09T00:00:00</q1:messageDate>
<q1:subject>-- subject goes here --</q1:subject>
<q1:message>This is a sample message</q1:message>
<q1:activity internalId="39" type="calendarEvent" />
</record>
</add>
</soap:Body>
</soap:Envelope>
```

SOAP Response

```
<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:documentInfo xmlns:ns1="urn:messages_2017_1.platform.webservices.netsuite.com">
<ns1:nsId>WEBSERVICES_721410_040920088711404551689975949_f1a83e87c1bf0</ns1:nsId>
</ns1:documentInfo>
</soapenv:Header>
<soapenv:Body>
<addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<writeResponse>
<ns2:status isSuccess="true" xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com" />
<baseRef internalId="46" type="message" xsi:type="ns3:RecordRef"
xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com" />
</writeResponse>
</addResponse>
</soapenv:Body>
</soapenv:Envelope>
```

C#

```
private void addMessage()
{
this.login(true);

Message msg = new Message();

RecordRef activityref = new RecordRef();
activityref.internalId = "39";
activityref.type = RecordType.calendarEvent;
activityref.typeSpecified = true;
msg.activity = activityref;

RecordRef authorref = new RecordRef();
authorref.internalId = "-5";
authorref.type = RecordType.employee;
authorref.typeSpecified = true;
msg.author = authorref;

RecordRef recipientref = new RecordRef();
recipientref.internalId = "-5";
recipientref.type = RecordType.employee;
recipientref.typeSpecified = true;
msg.recipient = recipientref;

DateTime searchDate = new DateTime();
searchDate = DateTime.Now;
searchDate = DateTime.Parse(searchDate.ToString("dd/MM/yyyy"));
msg.messageDate = searchDate;
msg.messageDateSpecified = true;
msg.subject = "-- subject goes here --";
msg.message = "This is a sample message";
msg.incoming = true;

WriteResponse writeRes = _service.add(msg);
if (writeRes.status.isSuccess)
{
_out.writeLn("\nThe message " + msg.internalId + " has been added successfully");
}
else
{
_out.error(getStatusDetails(writeRes.status));
}
}
```

Transactions

Review the following for general information about SuiteTalk transactions:

- Usage Notes for Transaction Record Types
- Multiple Shipping Routes in Web Services

The following transaction record types are supported in SuiteTalk:

- Advanced Intercompany Journal Entry
- Assembly Build
- Assembly Unbuild
- Bin Putaway Worksheet
- Bin Transfer
- Budget
- Cash Refund
- Cash Sale
- Charge
- Check
- Credit Memo
- Customer Deposit
- Customer Payment
- Customer Refund
- Deposit
- Deposit Application
- Estimate/Quote
- Expense Report
- Inbound Shipment
- Intercompany Journal Entry
- Intercompany Transfer Order
- Inventory Adjustment
- Inventory Cost Revaluation
- Inventory Transfer
- Invoice
- Item Demand Plan
- Item Fulfillment
- Item Receipt
- Item Supply Plan
- Journal Entry
- Manufacturing Operation Task
- Opportunity
- Paycheck
- Paycheck Journal
- Purchase Order

- Requisition
- Return Authorization
- Sales Order
- Statistical Journal Entry
- Time Bill (Track Time)
- Transfer Order
- Usage
- Vendor Bill
- Vendor Credit
- Vendor Payment
- Vendor Return Authorization
- Work Order
- Work Order Close
- Work Order Completion
- Work Order Issue
- Multiple Shipping Routes in Web Services

Usage Notes for Transaction Record Types

Note the following when working with transactions:

- To maintain performance, 5,000 lines per transaction is the maximum limit for transactions submitted through synchronous web services. With asynchronous web services, the limit is 10,000 lines.
- There is a 200 record limit for web services transactions.
- If you have the Multiple Shipping Routes (MSR) feature turned on in your NetSuite account, and you want to enabled MSR on a specific transaction, see Multiple Shipping Routes in Web Services.
- The *rate* field on a sublist can be set without explicitly setting price level to **custom**. Even though NetSuite does not treat the price level as required to set the rate field, we recommend that users set the item price level to "-1" (custom) at the time they are setting the item rate.
- The 2011.2 and later endpoints include a change from previous endpoints to the currency field in the body of transactions. Endpoints prior to 2011.2 included the **currencyName** string type field. As of the 2011.2 endpoint, this field has been replaced by the **currency** RecordRef type field. If you upgrade to the 2011.2 endpoint or later, you must update any code that references **currencyName.**
- In the 2012.1 and later endpoints, you can set the currency value for some newly created transactions when the Multiple Currencies feature is enabled. These edits is supported for the following transactions: Cash Refund, Cash Sale, Credit Memo, Customer Deposit, Customer Payment, Customer Refund, Esitmate, Invoice, Item Receipt, Opportunity, Purchase Order, Return Authorization, Sales Order, Vendor Bill, Vendor Credit, Vendor Payment, Vendor Return Authorization. (You cannot edit this value in earlier endpoints.) The currency value for a transaction must be one of the currencies defined for the transaction's customer in the CustomerCurrencyList. See Customer and Multiple Currencies and Customers.

Advanced Intercompany Journal Entry

An advanced intercompany journal entry is a specialized record type available only in OneWorld accounts. An advanced intercompany journal entry records debits and credits to be posted to ledger accounts for transactions between an originating subsidiary and multiple receiving subsidiaries. When you use advanced intercompany journal entries, you can also change the transaction currency to any currency set up in your system. In an account that has the Multi-Book Accounting feature enabled, you can also use this record type to create book specific advanced intercompany journal entries.

In the user interface, you access this record as follows:

- Advanced intercompany journal entries: Go to Transactions > Financial > Make Advanced Intercompany Journal Entries. For help working with this record in the user interface, see the help topic Advanced Intercompany Journal Entries.
- Book specific advanced intercompany journal entries: Go to Transactions > Financial > Make Book Specific Advanced Intercompany Journal Entries. This form is available only to accounts that use Multi-Book Accounting. For help working with this record in the user interface, see the help topic Advanced Intercompany Journal Entries in Multi-Book Accounting.

The advanced intercompany journal entry record is defined in the tranGeneral XSD, where it is called AdvInterCompanyJournalEntry.

Supported Operations

The following operations can be used with the advanced intercompany journal entry record.

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's Advanced Intercompany Journal Entry reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Maximum Number of Lines

You must have at least four line items to submit an advanced intercompany journal entry transaction. The maximum number of lines per journal entry is 1,000 when using synchronous web services processing. With asynchronous web services processing, the limit is 10,000 lines. If your data includes journal entries with more than 1,000 lines, CSV import can also serve as an alternative. For journal entry transactions submitted through CSV import, the limit is 10,000 lines per transaction.

Searching for Advanced Intercompany Journal Entries

You can use the advancedIntercompany boolean search field available both in the TransactionSearch and the AccountTransactionSearch search objects to search for advanced intercompany journal entries.

Sample Code

For examples of how to structure your code, refer to the code samples in Journal Entry Code Samples and Intercompany Journal Entry Code Samples.

Assembly Build

An assembly item is an inventory item made of several components, but identified as a single item. This type of item lets you define the members of an assembly and to separately track both the component items and the assembled items in inventory. For each assembly build, the assembly item stock level increases and the member items' individual stock levels decrease.

An assembly build transaction records the physical manufacture of an assembly item from component items and the related inventory level changes.

The assembly build record is defined in the tranInvt (inventory) XSD.

Supported Operations

The following operations can be used with the assembly build record.

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s assembly build reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.

- If this feature is not enabled, you do not need to use the Inventory Detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Assembly Unbuild

An assembly item is an inventory item made of several components, but identified as a single item. This type of item lets you define the members of an assembly and to separately track both the component items and the assembled items in inventory.

An assembly unbuild transaction records the physical taking apart of an assembly item back into its component items and the related inventory level changes. For each assembly unbuild, the assembly item stock level decreases and the member items' individual stock levels increase.

For details about this type of transaction, see the help topic Building Assembly Items. For details about this type of item, see the help topic Assembly Items.

The assembly unbuild record is defined in the tranInvt (inventory) XSD.

Supported Operations

The following operations can be used with the assembly unbuild record.

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s assembly unbuild reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Initializing Assembly Unbuilds

You can initialize an assembly unbuild from an Assembly Build.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the Inventory Detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Bin Putaway Worksheet

The Bin Putaway Worksheet transaction provides a list of bin numbers for items that need to be restocked. Items are added to this list any time an item that uses bins has a greater total on-hand count than the combined on-hand count of its associated bins. This transaction is available when the Bin Management feature is enabled. For more details, see the help topic Bin Putaway Worksheet.

The bin putaway worksheet record is defined in the tranInvt (inventory) XSD.

Supported Operations

The following operations can be used with bin putaway worksheet records:

get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s bin worksheet reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the Inventory Detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Bin Transfer

A Bin Transfer transaction moves items between bins within a warehouse. The transaction identifies the item, the source bin, the receiving bin, and the quantity to be moved. This type of transaction does not post to the chart of accounts and has no financial impact. The transfer only updates the quantity on hand in each bin for the items transferred.

This transaction is available when the Bin Management feature is enabled. For details, see Bin Transfers.

Note: Bin transfers can move items only if they are already in one or more bins.

The bin transfer record is defined in the tranInvt (inventory) XSD.

Supported Operations

The following operations can be used with bin transfer records:

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s bin transfer reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the Inventory Detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Budget

A budget records the expected values of income and expenses for your business. You can create budgets for specific customers, items, departments, classes, locations, or any combination of these criteria. You can also create budgets for multiple subsidiaries in NetSuite OneWorld. For general information about budgets, see the help topic Setting Up a Budget.

In web services, the budget record is defined in the tranFin (financial) XSD. Note that before updating a row on this record, you must perform a **get** on the entire record.

Supported Operations

The following operations can be used with the budget record.

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s budget reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Custom Segments for the Budget Record

You cannot use CustomFieldList to create custom fields on the budget record, but you can set a custom segment value for a budget record by setting its CustomFieldList property. You cannot create a custom segment, but you can use web services to add, delete, and update segment values.

You can use CustomJoin for BudgetSearch and BudgetSearchRow, which are both defined in the common XSD.

For details, see Working With Custom Segment Values, CustomFieldLists for Setting Custom Segment Values, and CustomFieldList.

Supported Accounting Periods in the Budget Record

The budget record supports more than 12 accounting periods for add, get, and search operations. If your accounting setup consists of more than 12 accounting periods, you can add, retrieve, and search for up to 24 accounting periods. For information about accounting periods, see the help topic Setting Up Accounting Periods.

Cash Refund

A cash refund transaction records the return of money to a customer who immediately paid for goods or services using cash, a check or a credit card.

For details about this type of transaction, see the help topic Refunding a Cash Sale.

The cash refund record is defined in the tranCust (customers) XSD.

Supported Operations

The following operations can be used with the cash refund record:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's cash refund reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Working with Credit Card Data

When working with credit card data, be aware of the following security features:

- When adding a transaction that uses a credit card number, you cannot identify the credit card number using a masked value such as ************5151. You must enter the full 16-digit number, or you can identify an existing credit card record through a RecordRef. (You can identify the full number using the **ccNumber** field. You can reference an existing record using the **creditCard** field.)
- Searches do not work if they include the operator **is** or **isNot** in conjunction with the ccNumber field. The only search operators that can apply to this field are **empty** and **notEmpty**.

Initializing Cash Refunds

You can initialize a cash refund from a Cash Sale or a Return Authorization.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the Inventory Detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Cash Sale

A cash sale transaction records the sale of goods or services for which immediate payment is received at the time of delivery. Cash sale line items specify the goods and services sold and their sales amounts. The sum of all sales amounts plus any applicable tax equals the total amount paid for the transaction.

For more details about this transaction, see the help topic Cash Sales.

The cash sale record is defined in the tranSales (sales) XSD.

Supported Operations

The following operations can be used with cash sale records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s cash sale reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Working with Credit Card Data

When working with credit card data, be aware of the following security features:

- When adding a transaction that uses a credit card number, you cannot identify the credit card number using a masked value such as ************5151. You must enter the full 16-digit number, or you can identify an existing credit card record through a RecordRef. (You can identify the full number using the **ccNumber** field. You can reference an existing record using the **creditCard** field.)
- Searches do not work if they include the operator **is** or **isNot** in conjunction with the ccNumber field. The only search operators that can apply to this field are **empty** and **notEmpty**.

Initializing Cash Sales

You can initialize a cash sale from a Customer, Estimate/Quote, Opportunity, or Sales Order.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

Note: To initialize a cash sale from a sales order that is in the Pending fulfillment status, you must enable the Invoice in Advance of Fulfillment preference. You can enable this preference at Setup > Accounting > Preferences > Accounting Preferences (Administrator), on the Order Management subtab. After enabling this preference, use the login operation to create a new web services session. For more information, see the help topic Caching Behavior in Web Services.

Working with Cash Sale Sublists

The SuiteTalk Schema Browser includes all sublists associated with the cash sale record. See the following information for usage notes regarding specific cash sale sublists. Usage notes are not provided for every sublist type.

- CashSaleSalesTeamList
- CashSaleItemList
- Cash SaleItemCostList
- CashSaleExpCostList
- CashSaleTimeList
- GiftCertRedemptionList

CashSaleSalesTeamList

The CashSaleSalesTeam list defines the sales team for a specific cash sale. This list is only available when the Team Selling feature is enabled in the account.

On add, a single line item is entered by default when a sales rep is currently associated with the entity set in the entity field. When a sales group is currently associated with the entity, the list is populated by the group.

CashSaleItemList

The orderLine field in this sublist establishes the relationship with an existing sales order, if any. You must set a value for orderLine to populate the createdFromfield.

Cash SaleItemCostList

These fields map to the Billable Items subtab on the Item list of a cash sale record. This is where you define how and when to bill item costs back to a customer. Billing items back to customers enables you to purchase items and supplies for an order or job, and then bill the cost to the customer. The Bill Costs To Customers feature must be enabled to use this list.

CashSaleExpCostList

These fields map to the Billable Expenses subtab on the Item list of a cash sale record. This is where you define how and when to bill expenses back to a customer. The Bill Costs To Customers feature must be enabled to use this list.

CashSaleTimeList

These fields map to the Billable Time subtab on the Item list of a cash sale record. This is where you define how and when to bill time back to a customer. The Bill Costs To Customers feature must be enabled to use this list.

GiftCertRedemptionList

This sublist is available on the invoice, sales order, and cash sale records.

Promotion Code Validation

Note that validation of promotion code date range for a cash sale transaction generated from a sales order is different in web services than in the user interface. In the user interface, as long as sales order date was within the promotion code date range, cash sale can be generated without error, even if cash sale date is outside of promotion code date range. In web services, cash sale date must be within promotion code date range.

Associating Cash Sales with Opportunities

You can associate estimates, cash sales, sales orders, and invoices with opportunities. After a transaction other than an estimate is associated with an opportunity, the opportunity's status is automatically set to **Closed Won**. After an opportunity's status is set to **Closed Won**, it is no longer available to be selected on other cash sale, sales order, or invoice records.

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.

- If this feature is not enabled, you do not need to use the Inventory Detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Charge

The charge record is used to represent a single billable amount that a client must pay.

This record is defined in the tranCust (customers) XSD, where it is called *Charge*.

The charge record is available when the Charge-Based Billing feature is enabled at Setup > Company > Enable Features, on the Transactions subtab. When the feature is enabled, you can access the charge record by choosing Transactions > Customers > Create Charges > List. For details about working with this record manually, see the help topic Generating Charges.

Supported Operations

add | addList | delete | deleteList | get | getList | search | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's charge reference page. For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Code Sample

The following example shows how to create a charge record.

```
public void testCharge() throws Exception{
       c = new NLWsClient();
       Charge charge = new Charge();
       charge.setBillTo(mrr("115"));
       charge.setStage(ChargeStage._holdForBilling);
       charge.setAmount(3.0);
       charge.setChargeDate(Calendar.getInstance());
       charge.setChargeType(ChargeRuleType._fixedDate);
       charge.setRate("2");
       charge.setQuantity(2.0);
       charge.setAmount(16.0);
       charge.setBillingItem(mrr("117"));
       charge.setCurrency(mrr("1"));
       charge.setSalesOrder(mrr("167"));
       charge.setSalesOrderLine(mrr("1"));
```

```
                charge.setProjectTask(mrr("3307"));
                charge.setTimeRecord(mrr("3"));
                c.addRecord(charge);
}
```

Check

A check transaction creates and records a check used to pay an expense, records an expense paid in cash and not entered as a bill, or records a non-check debit transaction, such as a debit card transaction, ATM (automated teller machine) transaction, or EFT (electronic funds transfer) payment. A check transaction records an expense directly to your books by debiting the expense account specified in the transaction detail and crediting the bank account the check selected for the check.

For more details about this type of transaction, see the help topic Checking.

The check record is defined in the tranBank (bank) XSD.

Supported Operations

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's the check reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the Inventory Detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Credit Memo

A credit memo transaction decreases the amount a customer owes you. This type of transaction can be used to reverse a charge billed to a customer. If a customer receives a credit memo after having paid an invoice, this memo can be applied to any of the customer's open or future invoices.

For more details about this type of transaction, see the help topic Customer Credit Memos.

The credit memo record is defined in the tranCust (customers) XSD.

Supported Operations

The following operations can be used with credit memo records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s credit memo reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Initializing Credit Memos

You can initialize a credit memo from a Customer, an Invoice, or a Return Authorization.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web

services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the Inventory Detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Note: For sample C# code creating a credit memo using web services, see the SuiteAnswers article Create a Credit Memo in Web Services via C#.

Customer Deposit

A customer deposit transaction records the funds received when a customer makes an advance payment for an order. This payment is recorded in the general ledger as a liability until the goods or services are actually delivered, and does not affect the customer's accounts receivable balance. After the order is filled, the deposit is applied against the invoice.

For more details about this type of transaction, see the help topic Customer Deposits.

The customer deposit record is defined in the tranCust (customers) XSD.

Supported Operations

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s customer deposit reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Working with Credit Card Data

When working with credit card data, be aware of the following security features:

- When adding a transaction that uses a credit card number, you cannot identify the credit card number using a masked value such as ************5151. You must enter the full 16-digit number, or you can identify an existing credit card record through a RecordRef. (You can identify the full number using the **ccNumber** field. You can reference an existing record using the **creditCard** field.)

- Searches do not work if they include the operator **is** or **isNot** in conjunction with the ccNumber field. The only search operators that can apply to this field are **empty** and **notEmpty**.

Creating a Customer Deposit based on a Sales Order

As of the 2012.1 endpoint, the salesOrder field is exposed for customer deposit transactions, so that customer deposits can be based on other types of sales orders in addition to cash sales, typically from invoices. Note that only sales orders with the same customer as in the customer deposit can be used.

The following example shows the creation of a customer deposit based on a sales order:

HTTP Request

```
<platformMsgs:add
          xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:platformMsgs="urn:messages_2017_1.platform.webservices.netsuite.com"
          xmlns:s0="urn:customers_2017_1.transactions.webservices.netsuite.com"
          xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
         <platformMsgs:record xsi:type="s0:CustomerDeposit">
            <s0:customer internalId="199" />
            <s0:salesOrder internalId="7818" />
            <s0:payment>214.0</s0:payment>
         </platformMsgs:record>
      </platformMsgs:add>
```

HTTP Response

```
<addResponse 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="7828" type="customerDeposit" xsi:type="platformCore:RecordRef"

              xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com" />
         </writeResponse>
      </addResponse>
```

Customer Payment

A customer payment transaction records a customer payment and applies it to the appropriate invoice or cash sale, decreasing the amount due and tracking income.

For details about this type of transaction, Customer Payments.

The customer payment record is defined in the tranCust (customers) XSD.

Supported Operations

The following operations can be used with customer payment records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's customer payment reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Working with Credit Card Data

When working with credit card data, be aware of the following security features:

- When adding a transaction that uses a credit card number, you cannot identify the credit card number using a masked value such as ************5151. You must enter the full 16-digit number, or you can identify an existing credit card record through a RecordRef. (You can identify the full number using the **ccNumber** field. You can reference an existing record using the **creditCard** field.)
- Searches do not work if they include the operator **is** or **isNot** in conjunction with the ccNumber field. The only search operators that can apply to this field are **empty** and **notEmpty**.

Initializing Customer Payments

You can initialize a customer payment from a Customer or an Invoice.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

Working with AutoApply

If you specify the "total amount of the original payment" as the payment value in a customer payment, autoApply through web services fails when there has already been an amount applied previously on the payment. This process fails when autoApply is set to true, and an invoice with apply = true already exists in the customer payment.

To work around this, you must set apply=false for the invoice. The following is the recommended workflow:

GET: (get the existing customer payment)

```
<get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<baseRef internalId="2262" type="customerPayment" xsi:type="ns1:RecordRef" xmlns:ns1="urn:core_
2017_1.platform.webservices.netsuite.com"/>
```

```
</get>
```

RESPONSE: (the response will return all the invoices that were already applied)

```
<ns3:applyList>
<ns3:apply>
<ns3:apply>true</ns3:apply>
<ns3:doc>1929</ns3:doc>
<ns3:total>27.92</ns3:total>
<ns3:due>27.92</ns3:due>
<ns3:amount>27.92</ns3:amount>
</ns3:apply>
<ns3:apply>
<ns3:apply>true</ns3:apply>
<ns3:doc>1930</ns3:doc>
<ns3:total>27.92</ns3:total>
<ns3:due>27.92</ns3:due>
<ns3:amount>27.92</ns3:amount>
</ns3:apply>
</ns3:applyList>
```

UPDATE: (update the customer payment - set paymentAmount to the original amount, set autoApply = true, **AND** set apply = false on all the invoices that have been already applied)

```
<update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<record internalId="2262" xsi:type="ns1:CustomerPayment" xmlns:ns1="urn:customers_2017_1.transa
ctions.webservices.netsuite.com">
<ns1:customer internalId="176" xsi:type="ns2:RecordRef" xmlns:ns2="urn:core_2017_1.platform.web
services.netsuite.com"/>
<ns1:payment xsi:type="xsd:double">80.0</ns1:payment>
<ns1:autoApply xsi:type="xsd:boolean">true</ns1:autoApply>
<ns1:applyList replaceAll="false" xsi:type="ns1:CustomerPaymentApplyList">
<ns1:apply xsi:type="ns1:CustomerPaymentApply">
<ns1:apply xsi:type="xsd:boolean">false</ns1:apply>
<ns1:doc xsi:type="xsd:long">1929</ns1:doc>
</ns1:apply>
<ns1:apply xsi:type="ns1:CustomerPaymentApply">
<ns1:apply xsi:type="xsd:boolean">false</ns1:apply>
<ns1:doc xsi:type="xsd:long">1930</ns1:doc>
</ns1:apply>
</ns1:applyList>
</record>
</update>
```

GET: (perform a get to verify that it worked)

```
<get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<baseRef internalId="2262" type="customerPayment" xsi:type="ns1:RecordRef" xmlns:ns1="urn:core_
2017_1.platform.webservices.netsuite.com"/>
</get>
```

RESPONSE: (this response shows that the update worked. The invoices that were applied before are still applied, and the rest of the Payment Amount was applied to new invoices)

```
<ns3:applyList>
```

```
<ns3:apply>
<ns3:apply>true</ns3:apply>
<ns3:doc>1933</ns3:doc>
<ns3:total>27.92</ns3:total>
<ns3:due>27.92</ns3:due>
<ns3:amount>24.16</ns3:amount>
</ns3:apply>
<ns3:apply>
<ns3:apply>true</ns3:apply>
<ns3:doc>1929</ns3:doc>
<ns3:total>27.92</ns3:total>
<ns3:due>27.92</ns3:due>
<ns3:amount>27.92</ns3:amount>
</ns3:apply>
<ns3:apply>
<ns3:apply>true</ns3:apply>
<ns3:doc>1930</ns3:doc>
<ns3:total>27.92</ns3:total>
&lt...
```

Setting Accounts on Customer Payments

The 2013.1 endpoint changed the precedence for setting the account on customer payments from earlier endpoints. For the 2013.1 and later endpoints, if Undep. Funds is set to True, this value overrides Payment Method, and if Undep. Funds is set to False, Account overrides Payment Method.

In the 2012.2 and earlier endpoints, the account for a customer payment can only be set to Undep. Funds if the Payment Method and Account fields do not have values set, and there are also further complexities. Review the following table for details about setting the account on a customer payment in these earlier endpoints:

Payment Method Group with Undep. Funds or Deposit to Account?	Is Payment Method Field Value Set?	What Is Undep. Funds Field Value?	Is Account Field Value Set?	What is Result?
Deposit to Account	Yes	False	Yes	account = value set for Account Field
Deposit to Account	Yes	True	No	account = account associated with selected payment meethod
Group with Undep. Funds	No	False	Yes	FAIL
Group with Undep. Funds	No	True	No	account = Undeposited Funds account
Group with Undep. Funds	No	True	Yes	FAIL
Deposit to Account	Yes	True	Yes	account = value set for Account Field

Customer Refund

A customer refund transaction records the return of funds to a customer who paid for goods or services using cash, a check, or a credit card. The refund is generally made in cash or by check.

For details about this type of transaction, see the help topic Customer Credits and Refunds.

The customer refund record is defined in the tranCust (customers) XSD.

Supported Operations

The following operations can be used to modify customer refund records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's customer refund reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Working with Credit Card Data

When working with credit card data, be aware of the following security features:

- When adding a transaction that uses a credit card number, you cannot identify the credit card number using a masked value such as ************5151. You must enter the full 16-digit number, or you can identify an existing credit card record through a RecordRef. (You can identify the full number using the **ccNumber** field. You can reference an existing record using the **creditCard** field.)
- Searches do not work if they include the operator **is** or **isNot** in conjunction with the ccNumber field. The only search operators that can apply to this field are **empty** and **notEmpty**.

Initializing Customer Refunds

You can initialize a customer refund from a Customer or a Credit Memo.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

Deposits Saved in CustomerRefundApplyList

If you add a customer refund with deposits in the CustomerRefundDepositList sublist, note that the deposits are actually saved in the CustomerRefundApplyList sublist. The following sample SOAP code illustrates this case.

Customer Refund Add Request

```
<add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record externalId="Pear16027841" xsi:type="ns8:CustomerRefund" xmlns:ns8="urn:cust
omers_2017_1.transactions.webservices.netsuite.com">
               <ns8:customer internalId="3023" xsi:type="ns9:RecordRef" xmlns:ns9="urn:core_201
7_1.platform.webservices.netsuite.com">
                  <ns9:name xsi:type="xsd:string">Strawberry13167584</ns9:name>
               </ns8:customer>
truncated portion
               <ns8:depositList replaceAll="false" xsi:type="ns8:CustomerRefundDepositList">
                  <ns8:customerRefundDeposit xsi:type="ns8:CustomerRefundDeposit">
                     <ns8:apply xsi:type="xsd:boolean">true</ns8:apply>
                     <ns8:doc xsi:type="xsd:long">4354</ns8:doc>
                     <ns8:depositDate xsi:type="xsd:dateTime">2012-07-25T07:00:00.000Z</ns8:dep
ositDate>
                     <ns8:refNum xsi:type="xsd:string">26</ns8:refNum>
                     <ns8:total xsi:type="xsd:double">40.0</ns8:total>
                     <ns8:remaining xsi:type="xsd:double">40.0</ns8:remaining>
                  </ns8:customerRefundDeposit>
               </ns8:depositList>
            </record>
         </add>
```

Response to Later Get Request

```
<getResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <readResponse>
               <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
               <record internalId="4355" externalId="Pear16027841" xsi:type="tranCust:CustomerR
efund" xmlns:tranCust="urn:customers_2017_1.transactions.webservices.netsuite.com">
                  <tranCust:customer internalId="3023" xmlns:platformCore="urn:core_2017_1.plat
form.webservices.netsuite.com">
                     <platformCore:name>Strawberry13167584</platformCore:name>
                  </tranCust:customer>
truncated portion
                  <tranCust:applyList>
                     <tranCust:apply>
                        <tranCust:apply>true</tranCust:apply>
                        <tranCust:doc>4356</tranCust:doc>
                        <tranCust:line>1</tranCust:line>
                        <tranCust:applyDate>2012-07-25T00:00:00.000-07:00</tranCust:applyDate>
                        <tranCust:type>Deposit Application</tranCust:type>
                        <tranCust:total>40.0</tranCust:total>
                        <tranCust:due>40.0</tranCust:due>
```

```
                    <tranCust:currency>USA</tranCust:currency>
                    <tranCust:amount>40.0</tranCust:amount>
                </tranCust:apply>
            </tranCust:applyList>
        </record>
    </readResponse>
</getResponse>
```

Deposit

You use the deposit record to adjust the balance of an account.

This record is defined in tranBank XSD, where it is called deposit.

In the UI, you access this record at Transactions > Bank > Make Deposits. For help working with this record in the user interface, see the help topic Making Deposits.

For more details on using SuiteTalk to work with the deposit record, see the following sections:

- Deposit Supported Operations
- Deposit Usage Notes and Field Summary
- Deposit Code Samples
- Common Errors With Deposit Records

Deposit Supported Operations

The following operations are supported for use with the deposit record:

add | addList | delete | deleteList | get | getList |getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Important: In the 2015.2 and later WSDLs, the behavior of the **add** operation, when used with the deposit record, is slightly changed compared with previous endpoints. With previous WSDLs, the add operation respected the user-specific filters saved on the Deposit form, on the Deposits > Payments subtab. In the UI, if you set these filters and return later to the page, your selections for these filters are still used. The 2015.1 and earlier WSDLs respect these filters. However, the 2015.2 and later endpoints do not respect these choices. With these endpoints, no filters are applied during an add request.

Deposit Usage Notes and Field Summary

The purpose of the deposit record is to increase the balance of an account. To create a record, you must identify the account to credit and the source of deposit, as follows:

- Use the account body field to specify the account that receives the deposit.
- Use the Payments and Other Deposits sublists to identify the source of the deposit.

You can optionally use another sublist, Cash Back, to credit other accounts besides the one named in the account body field. When you use this approach, the net value of the deposit is lowered by the amount of Cash Back lines. For example, suppose you include one Payments sublist item of $500, one Other Deposit line of $500, and one Cash Back line of $100. The net value of the deposit is $900.

Every deposit record must have at least one line in one of the sublists described in the following table. Each line must include an amount, and the net sum of all line amounts must be positive.

The following table shows additional details about working with each sublist.

Name	Keyed?	Notes
Cash Back	No	For each line, you identify an account to be credited. Note that the amount field in this sublist can take a negative value, which you would enter if you wanted to debit the account. If you use this approach, the system saves the Cash Back line as a line in the Other Deposits sublist.
Other Deposits	No	For each line, you must identify an account to be debited. You can optionally populate the Name field by referencing an existing entity record. The amount of each line must be positive.
Payments	Yes, the key is the internal ID of the transaction represented by the line.	Each line must reference an existing record for a transaction, such as a payment or cash sale, that posted to the Undeposited Funds account. You can find valid transactions at Transactions > Bank > Make Deposits. Review the transactions listed on the Deposits subtab under Payments.

For an example of working with the Cash Back and Other Deposits sublists, see Adding a Record With Other Deposit, Cash Back Lines. For an example of working with the Payments sublist, see Adding a Record With Payment Lines and Updating the Deposit Record.

If you use the Multi-Book Accounting feature, be aware that the Accounting Book Detail sublist is read-only and not scriptable.

More Information

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s deposit reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Deposit Code Samples

For examples of how to work with deposit records using SuiteTalk, refer to the following sections:

- Adding a Record With Payment Lines
- Adding a Record With Other Deposit, Cash Back Lines

- Updating the Deposit Record

Adding a Record With Payment Lines

This example shows how to add a deposit record using the Payments sublist.

Each line in the Payments sublist must reference an existing transaction that was posted to the Undeposited Funds account. The status of the transaction must either be Undeposited or the transaction can be undefined. The status cannot be Deposited.

You can view a list that includes valid transactions in the user interface. Open the standard entry form for the record, at Transactions > Bank > Make Deposits. Navigate to the Payments subtab, which is on the Deposits subtab. The Payments subtab shows all transactions available to be used in the sublist.

In the following example, a new record is created with three lines in the Payments sublist.

C#

```
private void addDeposit()
{

   // Create object.

   Deposit myDeposit = new Deposit();


   // Set an external Id.

   myDeposit.externalId = "804A";


   // Identify the account that will receive the deposit.

   RecordRef myAccount = new RecordRef();
   myAccount.internalId = "6";
   myDeposit.account = myAccount;


   // Create an instance of the Payment sublist that can take as many as three lines.

   myDeposit.paymentList  = new DepositPaymentList();
   myDeposit.paymentList.depositPayment = new DepositPayment[3];


   // Populate the lines using existing payment records that posted to the
   // Undeposited Funds account. The id field points to the internal ID
   // of the payment record.

   myDeposit.paymentList.depositPayment[0] = new DepositPayment();
   myDeposit.paymentList.depositPayment[0].deposit = true;
   myDeposit.paymentList.depositPayment[0].depositSpecified = true;
   myDeposit.paymentList.depositPayment[0].id = 1969;
   myDeposit.paymentList.depositPayment[0].idSpecified = true;

   myDeposit.paymentList.depositPayment[1] = new DepositPayment();
   myDeposit.paymentList.depositPayment[1].deposit = true;
   myDeposit.paymentList.depositPayment[1].depositSpecified = true;
   myDeposit.paymentList.depositPayment[1].id = 1995;
   myDeposit.paymentList.depositPayment[1].idSpecified = true;

   myDeposit.paymentList.depositPayment[2] = new DepositPayment();
   myDeposit.paymentList.depositPayment[2].deposit = true;
   myDeposit.paymentList.depositPayment[2].depositSpecified = true;
   myDeposit.paymentList.depositPayment[2].id = 1671;
   myDeposit.paymentList.depositPayment[2].idSpecified = true;


   // Execute the operation.

   _service.add(myDeposit);

}
```

SOAP Request

```
<soap:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record xmlns:q1="urn:bank_2017_1.transactions.webservices.netsuite.com" xsi:type="q1:Dep
osit" externalId="804A">
         <q1:account internalId="6" />
         <q1:paymentList>
            <q1:depositPayment>
               <q1:deposit>true</q1:deposit>
               <q1:id>1969</q1:id>
         </q1:depositPayment>
            <q1:depositPayment>
               <q1:deposit>true</q1:deposit>
               <q1:id>1995</q1:id>
            </q1:depositPayment>
            <q1:depositPayment>
               <q1:deposit>true</q1:deposit>
               <q1:id>1671</q1:id>
            </q1:depositPayment>
            </q1:paymentList>
      </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 internalId="7851" externalId="804A" type="deposit" xsi:type="platformCore:Rec
ordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
```

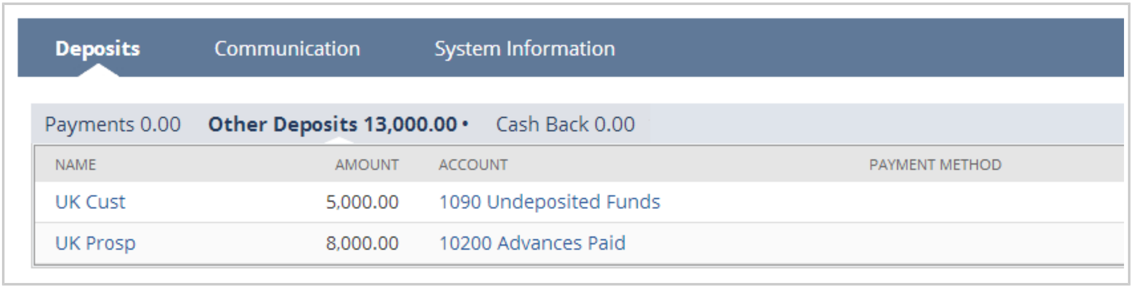
```
    </addResponse>
</soapenv:Body>
```

Adding a Record With Other Deposit, Cash Back Lines

In some cases, you might want to create a deposit record using the Other Deposits and Cash Back sublists. The Other Deposits sublist is used to debit one or more accounts named in the sublist lines. By contrast, the Cash Back list is generally used to credit accounts.

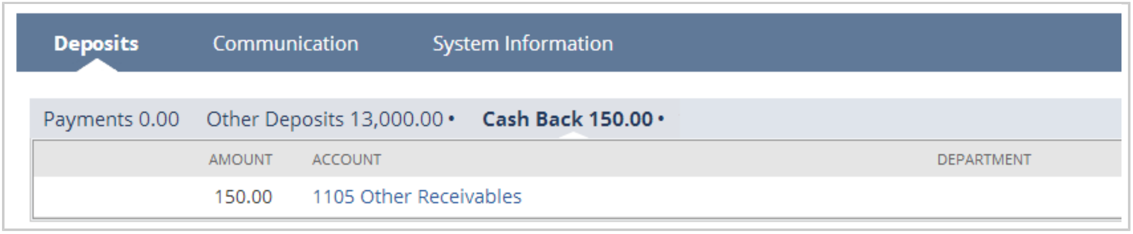
Each sublist has an amount field, but these fields are different. That is, in the Other Deposits sublist, a positive amount field debits the account identified by the line. In the Cash Back sublist, the amount field credits the account identified by the line.

For example, in the illustration below, the Undeposited Funds account is debited $5,000 and the Advances Paid account is debited $8,000.



NAME	AMOUNT	ACCOUNT	PAYMENT METHOD
UK Cust	5,000.00	1090 Undeposited Funds	
UK Prosp	8,000.00	10200 Advances Paid	

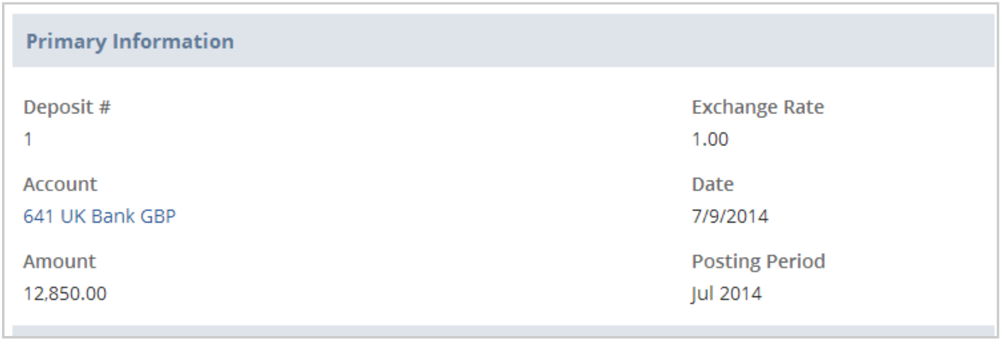
In the illustration below, the Other Receivables account is credited $150.



AMOUNT	ACCOUNT	DEPARTMENT
150.00	1105 Other Receivables	

In this example, the net sum of the lines is a positive amount ($12,850), which means the record can be added without error. (When the net value of all sublist lines is negative, the record cannot be added.)

The recipient of the funds is identified by the account body field. For example, in the screenshot below, the recipient is the UK Bank GBP account.



Deposit #	Exchange Rate
1	1.00
Account	Date
641 UK Bank GBP	7/9/2014
Amount	Posting Period
12,850.00	Jul 2014

The following code sample shows how you would create this record.

C#

```
private void addSecondDeposit()
{

   // Create object.

   Deposit myDeposit = new Deposit();


   // Set an external Id.

   myDeposit.externalId = "805A";


   // Identify the account that will receive the deposit.

   RecordRef myAccount = new RecordRef();
   myAccount.internalId = "6";
   myDeposit.account = myAccount;


   // Create an instance of the Other Deposits sublist that can take up to two lines.

   myDeposit.otherList = new DepositOtherList();
   myDeposit.otherList.depositOther = new DepositOther[2];


   // Populate the lines.

   RecordRef myEntity = new RecordRef();
   myEntity.type = RecordType.customer;
   myEntity.typeSpecified = true;
   myEntity.internalId = "1766";

   RecordRef myOtherDepositAccount = new RecordRef();
   myOtherDepositAccount.internalId = "12";

 myDeposit.otherList.depositOther[0] = new DepositOther();
 myDeposit.otherList.depositOther[0].entity = myEntity;
   myDeposit.otherList.depositOther[0].amount = 5000;
   myDeposit.otherList.depositOther[0].amountSpecified = true;
 myDeposit.otherList.depositOther[0].account = myOtherDepositAccount;


   RecordRef myEntity2 = new RecordRef();
   myEntity2.type = RecordType.customer;
   myEntity2.typeSpecified = true;
   myEntity2.internalId = "1974";

   RecordRef myOtherDepositAccount2 = new RecordRef();
   myOtherDepositAccount2.internalId = "244";
   myDeposit.otherList.depositOther[1] = new DepositOther();
   myDeposit.otherList.depositOther[1].entity = myEntity2;
   myDeposit.otherList.depositOther[1].amount = 8000;
   myDeposit.otherList.depositOther[1].amountSpecified = true;
   myDeposit.otherList.depositOther[1].account = myOtherDepositAccount2;


   // Create an instance of the Cash Back sublist.

   myDeposit.cashBackList = new DepositCashBackList();
   myDeposit.cashBackList.depositCashBack = new DepositCashBack[1];


   // Populate the sublist.

   RecordRef myCashBackAccount = new RecordRef();
   myCashBackAccount.internalId = "9";

   myDeposit.cashBackList.depositCashBack[0] = new DepositCashBack();
   myDeposit.cashBackList.depositCashBack[0].amount = 150;
   myDeposit.cashBackList.depositCashBack[0].amountSpecified = true;
   myDeposit.cashBackList.depositCashBack[0].account = myCashBackAccount;


  // Execute the operation.

   _service.add(myDeposit);

}
```

SOAP Request

```
<soap:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record xmlns:q1="urn:bank_2017_1.transactions.webservices.netsuite.com" xsi:type="q1:Dep
osit" externalId="805A">
         <q1:account internalId="6" />
            <q1:otherList>
               <q1:depositOther>
                  <q1:entity internalId="1766" type="customer" />
                  <q1:amount>5000</q1:amount>
                  <q1:account internalId="12" />
               </q1:depositOther>
               <q1:depositOther>
                  <q1:entity internalId="1974" type="customer" />
                  <q1:amount>8000</q1:amount>
                  <q1:account internalId="244" />
               </q1:depositOther>
            </q1:otherList>
            <q1:cashBackList>
            <q1:depositCashBack>
               <q1:amount>150</q1:amount>
               <q1:account internalId="9" />
            </q1:depositCashBack>
         </q1:cashBackList>
      </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 internalId="7952" externalId="805A" type="deposit" xsi:type="platformCore:Rec
ordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Updating the Deposit Record

The following example shows how to update the deposit record by removing a line from the Payments sublist and adding a different line. Also in this example, the Cash Back sublist is overwritten with new entries. The Cash Back sublist is not keyed, so overwriting it is the only way to update it.

Because the Payments sublist is basically a list of other records, the only change you can make to an existing line is to remove it from the sublist.

C#

```
private void updateDeposit()
{

   // Create object.

   Deposit myUpdatedDeposit = new Deposit();


   // Identify the record to update.

   myUpdatedDeposit.externalId = "804A";


   // Create an instance of the Payment sublist. Set ReplaceAll
   // to false, since you want to selectively update the list.

   myUpdatedDeposit.paymentList = new DepositPaymentList();
   myUpdatedDeposit.paymentList.depositPayment = new DepositPayment[2];
   myUpdatedDeposit.paymentList.replaceAll = false;


   // Identify the changes you want to make. To remove an existing line
   // from the sublist, set the deposit field to false. To add a new line,
   // set the field to true. The id field identifies the internal ID of the
   // transaction record being deposited


   myUpdatedDeposit.paymentList.depositPayment[0] = new DepositPayment();
   myUpdatedDeposit.paymentList.depositPayment[0].id = 1969;
   myUpdatedDeposit.paymentList.depositPayment[0].idSpecified = true;
   myUpdatedDeposit.paymentList.depositPayment[0].deposit = false;
   myUpdatedDeposit.paymentList.depositPayment[0].depositSpecified = true;

   myUpdatedDeposit.paymentList.depositPayment[1] = new DepositPayment();
   myUpdatedDeposit.paymentList.depositPayment[1].id = 1742;
   myUpdatedDeposit.paymentList.depositPayment[1].idSpecified = true;
   myUpdatedDeposit.paymentList.depositPayment[1].deposit = true;
   myUpdatedDeposit.paymentList.depositPayment[1].depositSpecified = true;


   // Create an instance of the Cash Back sublist.

   myUpdatedDeposit.cashBackList = new DepositCashBackList();
   myUpdatedDeposit.cashBackList.depositCashBack = new DepositCashBack[1];


   // Populate the sublist.

   RecordRef myCashBackAccount = new RecordRef();
   myCashBackAccount.internalId = "38";

   myUpdatedDeposit.cashBackList.depositCashBack[0] = new DepositCashBack();
   myUpdatedDeposit.cashBackList.depositCashBack[0].amount = 1500;
   myUpdatedDeposit.cashBackList.depositCashBack[0].amountSpecified = true;
   myUpdatedDeposit.cashBackList.depositCashBack[0].account = myCashBackAccount;


   // Execute the operation.

   _service.update(myUpdatedDeposit);

}
```

SOAP Request

```
<soap:Body>
   <update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record xmlns:q1="urn:bank_2017_1.transactions.webservices.netsuite.com" xsi:type="q1:Dep
osit" externalId="804A">
         <q1:paymentList replaceAll="false">
            <q1:depositPayment>
               <q1:deposit>false</q1:deposit>
               <q1:id>1969</q1:id>
            </q1:depositPayment>
            <q1:depositPayment>
               <q1:deposit>true</q1:deposit>
               <q1:id>1742</q1:id>
            </q1:depositPayment>
            </q1:paymentList>
            <q1:cashBackList>
               <q1:depositCashBack>
                  <q1:amount>1500</q1:amount>
                  <q1:account internalId="38" />
               </q1:depositCashBack>
            </q1:cashBackList>
         </record>
   </update>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
   <updateResponse 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 internalId="7851" externalId="804A" type="deposit" xsi:type="platformCore:Rec
ordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </updateResponse>
</soapenv:Body>
```

Common Errors With Deposit Records

When you are working with deposit records, the following failure notices can occur in your SOAP responses.

Invalid id reference key x

When seen in conjunction with the Payment sublist's id field, double-check that the id you used matches the internal ID of a transaction that posted to the Undeposited Funds account. Also, the transaction's status cannot be Deposited. The status must be Undeposited, or the status can be undefined.

The total can not be negative

This error can occur if the net value of your sublist line amounts is negative. The total amount of the deposit must be positive. Remember that a positive amount in the Cash Back list reduces the amount of the deposit.

Deposit Application

A deposit application transaction applies a customer deposit against an invoice after the order is complete.

For details about this type of transaction, see the help topic Applying a Customer Deposit.

The deposit application record is defined in tranCust (customers) XSD.

Supported Operations

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's deposit application reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Initializing Deposit Applications

You can initialize a Deposit Application from a Customer Deposit.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

Initializing Deposit Applications from Customer Deposits

For NetSuite accounts with multiple A/R accounts, the InitializeAuxRefType arAccount parameter can be used to specify an A/R account a deposit application record is initialized from a customer deposit record. The returned record contains lines that correspond to the specified A/R account.

The arAccount parameter is defined in InitializeAuxRefType in the coreTypes XSD.

For information about working with the customer deposit record in the UI, see the help topic Customer Deposits. For information about working with customer deposit records in web services, see Customer Deposit.

The following example shows the initialization of a deposit application from a customer deposit:

Java

```
DepositApplication da = (DepositApplication)c.initialize(new InitializeRef(null, InitializeRefT
ype.customerDeposit, “1506”, null), InitializeType.depositApplication, new InitializeAuxRef(nul
l, InitializeAuxRefType.arAccount, “9”, null));

System.out.println(da.getArAcct());

da.getApplyList().getApply(0).setApply(true);
```

```
c.addRecord(da);
```

SOAP Request

```
<soapenv:Body>
        <initialize xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <initializeRecord>
                <ns9:type xmlns:ns9="urn:core_2017_1.platform.webservices.netsuite.com">deposit
Application</ns9:type>
                <ns10:reference type="customerDeposit" internalId="1506" xmlns:ns10="urn:core_2
017_1.platform.webservices.netsuite.com"/>
                <ns11:auxReference type="arAccount" internalId="9" xmlns:ns11="urn:core_2017_1.
platform.webservices.netsuite.com"/>
            </initializeRecord>
        </initialize>
    </soapenv:Body>
```

SOAP Response

```
<soapenv:Body>
        <initializeResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <readResponse>
                <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platf
orm.webservices.netsuite.com"/>
                <record xsi:type="tranCust:DepositApplication" xmlns:tranCust="urn:customers_20
17_1.transactions.webservices.netsuite.com">
                    <tranCust:createdDate>2016-11-07T09:35:00.000-08:00</tranCust:createdDate>
                    <tranCust:lastModifiedDate>2016-11-07T09:35:00.000-08:00</tranCust:lastModi
fiedDate>
                    <tranCust:arAcct internalId="9" xmlns:platformCore="urn:core_2017_1.platfor
m.webservices.netsuite.com">
                        <platformCore:name>11200 Delinquent Accounts</platformCore:name>
                    </tranCust:arAcct>
                    <tranCust:status>Not Deposited</tranCust:status>
                    <tranCust:customer internalId="87" xmlns:platformCore="urn:core_2017_1.plat
form.webservices.netsuite.com">
                        <platformCore:name>Abe Simpson</platformCore:name>
                    </tranCust:customer>
                    <tranCust:tranDate>2016-11-07T00:00:00.000-08:00</tranCust:tranDate>
                    <tranCust:postingPeriod internalId="288" xmlns:platformCore="urn:core_2017_
1.platform.webservices.netsuite.com">
                        <platformCore:name>Nov 2016</platformCore:name>
                    </tranCust:postingPeriod>
                    <tranCust:deposit internalId="1506" xmlns:platformCore="urn:core_2017_1.pla
tform.webservices.netsuite.com">
                        <platformCore:name>Customer Deposit #1</platformCore:name>
                    </tranCust:deposit>
                    <tranCust:total>222.0</tranCust:total>
                    <tranCust:applied>0.0</tranCust:applied>
                    <tranCust:unapplied>222.0</tranCust:unapplied>
                    <tranCust:tranId>To Be Generated</tranCust:tranId>
                </record>
            </readResponse>
        </initializeResponse>
    </soapenv:Body>
```

Estimate/Quote

An estimate transaction, sometimes referred to as a quote, is a non-posted record of estimated charges to a customer. An estimate can be printed, emailed, or faxed to the customer. After the customer accepts the estimate, it can be converted into a sales order, invoice, or cash sale. This transaction is available when the Estimates feature is enabled at Setup > Company > Enable Features, on the Transactions subtab.

Note: Estimates have no accounting impact until they are converted into invoices or cash sales.

For more details about this type of transaction, see the help topic Estimates.

The estimate/quote record is defined in the tranSales (sales) XSD.

Supported Operations

The following operations can be used with estimate records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's estimate reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Initializing Estimates

You can initialize an estimate from an Opportunity.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

Accessing Serial/Lot Number Data for Line Items

As of the 2011.2 endpoint, code to access serial number or lot number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the Inventory Detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

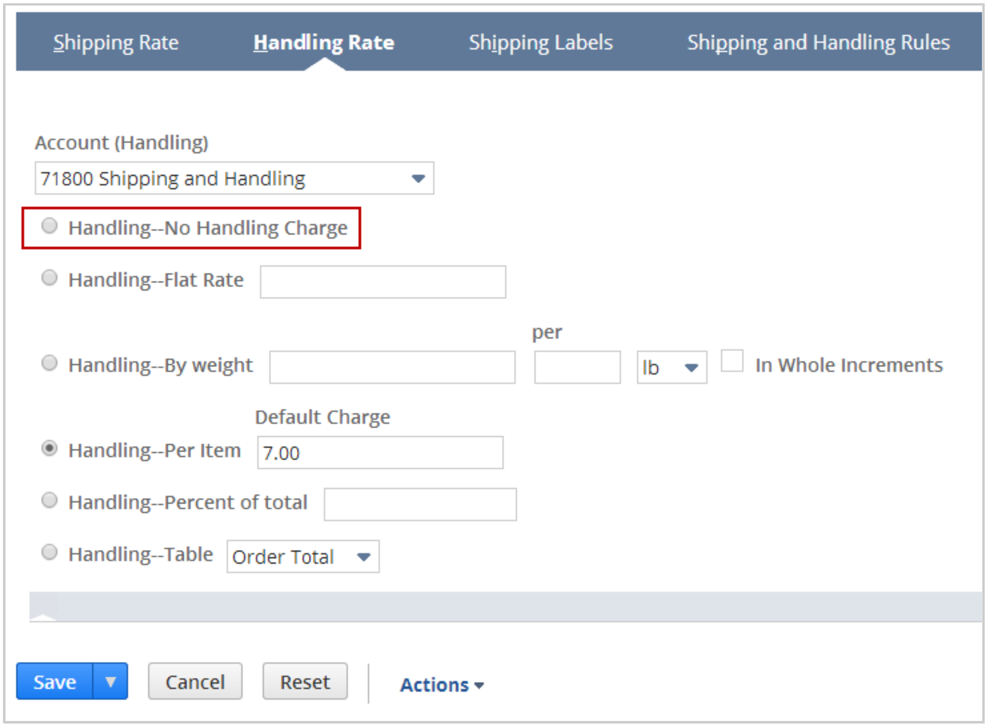
Setting Handling Cost

If you want to set a value for handlingCost, be aware of the following guidelines.

Only Certain Shipping Methods Allow Handling Cost To Be Set

In both the UI and in web services, the default behavior of the system is that you cannot specify a handling cost on a new estimate. To be able to enter a value for handling cost, you must choose a shipping method that enables the handling cost field.

You can identify an appropriate shipping method by reviewing the details for the corresponding shipping item. Choose Lists > Accounting > Shipping Items, then select the shipping item you want to review (each shipping item corresponds with a shipping method). In the item record, review the details on the Handling Rate subtab. As long as the Handling - No Handling Charge is **not** selected, then choosing this shipping item lets you set a handlingCost value.



Account (Handling)
71800 Shipping and Handling

- Handling--No Handling Charge
- Handling--Flat Rate
- Handling--By weight ___ per ___ lb ☐ In Whole Increments
- [✓ Handling--Per Item] Default Charge 7.00
- Handling--Percent of total
- Handling--Table Order Total

During an add operation, you must set the shipMethod value in the same SOAP request where you set the handlingCost value. If you fail to set any value for shipMethod, or if you choose a value that does not enable the handling cost field, the system generates an error reading in part “You do not have permissions to set a value for element handlingcost.”

You Must Specify a Shipping Cost

If you are doing an add operation, be aware that in some cases the system may overwrite the handlingCost value specified in your SOAP request. To avoid this, make sure that your SOAP request include a shippingCost value. If you fail to include a shippingCost, the system overwrites your handlingCost value with an automatically generated value.

This behavior is specific to the add operation. During an update, it is possible to set a value for handlingCost without specifying the shippingCost.

Expense Report

An expense report transaction records an employee's expenses for approval and conversion into a bill. The expense total remains in an unapproved expense account and has no accounting impact until the expense is approved by someone with accounting authority. After an expense report is approved, a bill is created and the expense amount is reflected on the books.

This transaction is available when the Estimates feature is enabled at Setup > Company > Enable Features, on the Employees subtab.

For more details about this type of transaction, see the help topic Expense Reporting.

The expense report record is defined in the tranEmp (employees) XSD.

Supported Operations

The following operations can be used with expense report records:

add | addList |attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Note: Attach / detach is supported for file attachments.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s expense report reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

When working with the Expenses sublist, be aware of the following: The Rate and Quantity fields are available only for line items where the expense category record has the rateRequired field set to true. For more details on working with expense categories, see Expense Category and Understanding Expense Categories.

Inbound Shipment

The inbound shipment record is available when the Inbound Shipment Management feature is enabled under Shipping and Receiving on the Transactions subtab at Setup > Company > Setup Tasks > Enable Features.

In the user interface, you can access the inbound shipment record at Transactions > Purchases > Create Inbound Shipment.

For information about working with this record in the user interface, see the help topic Inbound Shipment Management.

The inbound shipment record is defined in the tranPurch XSD.

Supported Operations

The following operations can be used with inbound shipment records:

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s Inbound Shipment reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Intercompany Journal Entry

An intercompany journal entry record is a specialized type of record available only in OneWorld accounts. An intercompany journal entry records debits and credits to be posted to ledger accounts for transactions between two subsidiaries. In an account that has the Multi-Book Accounting feature enabled, you can also use this record type to create book specific intercompany journal entries.

In the user interface, you access this record as follows:

- **Intercompany journal entries** – Go to Transactions > Financial > Make Intercompany Journal Entries. For help working with this record in the user interface, see the help topic Intercompany Journal Entries.
- **Book specific intercompany journal entries** – Go to Transactions > Financial > Make Book Specific Intercompany Journal Entries. Note that this form is available only to accounts that use Multi-Book Accounting. For help working with this record in the user interface, see the help topic Intercompany Journal Entries in Multi-Book Accounting.

The intercompany journal entry record is defined in the tranGeneral XSD, where it is called InterCompanyJournalEntry.

For more information about using SuiteTalk to work with the intercompany journal entry record, see the following sections:

- Intercompany Journal Entry Supported Operations
- Intercompany Journal Entry Body and Sublist Fields
- Intercompany Journal Entry Code Samples
- Common Errors With Intercompany Journal Entries

Intercompany Journal Entry Supported Operations

The following operations can be used with the intercompany journal entry record.

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

For examples of how to use the add operation, see Adding an Intercompany Journal Entry and Adding a Book Specific Intercompany Journal Entry.

Intercompany Journal Entry Body and Sublist Fields

Be aware that the way you interact with certain fields varies depending on whether the intercompany journal entry is book specific. For more information, see the following sections:

- Body Fields
- Sublist Fields

Note: For a full list of available fields, refer to the intercompany journal entry reference page in the SuiteTalk Schema Browser. For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Body Fields

The following table describes several key fields on the intercompany journal entry record. The way you use these fields varies depending on whether the journal entry is book specific. The guidance in this table assumes that you are adding a new intercompany journal entry record.

Field	Label in UI	To create a regular intercompany journal entry (not book specific)	To create a book specific intercompany journal entry
accountingBook	Accounting Book	Do not use this field. (Using this field automatically makes the journal entry book specific.)	Use this field to name the accounting book that this journal entry is for. You must choose an accounting book that has at least two subsidiaries listed on its Currencies subtab.
currency	Currency	You can omit a value for this field, and the system automatically uses the base currency for the subsidiary named by the subsidiary field. Alternatively, you can name the base currency for the subsidiary identified by the toSubsidiary field. These two currencies are the only valid values.	
exchangeRate	Exchange Rate	This field reflects the exchange rate between: ■ The value in the intercompany journal entry's currency field (which is also the base currency for one of the subsidiaries identified in this transaction). ■ The base currency for the other subsidiary identified in the transaction. You can leave this field empty to let the system fill in the value, or you can specify an exchange rate.	
isBookSpecific	This field is hidden in the UI.	Omit a value for this field. The system automatically assigns the correct value to this field. (The correct value in this case is false.)	If you are actively setting a value for the accountingBook field, you can omit a value for isBookSpecific. The system will automatically assign this field the correct value. (The correct value in this case is true.) As an alternative, you can set isBookSpecific to true and omit a value for accounting book. This approach creates a book specific intercompany journal entry for the primary accounting book.
memo	Memo	This field is available only with the 2015.2 WSDL and later. By contrast, the Memo field from the Lines sublist is available in all supported WSDLs.	
subsidiary	Subsidiary	Identify the subsidiary initiating the transaction.	
toSubsidiary	To Subsidiary	Identify the second subsidiary involved in the transaction.	

Sublist Fields

Only the line sublist is exposed to web services. This sublist is the main sublist on the record. It is used to record credits and debits. Every intercompany journal entry must have at least four lines in this sublist: two pairs of credit and debit values. Within each pair, the credit value must equal the debit value.

Important: The maximum number of lines per intercompany journal entry is **1,000** when using synchronous web services processing. With asynchronous web services processing, the limit is 10,000 lines. If your import includes transactions with more than 1,000 lines, CSV import can also serve as an alternative. For transactions submitted through CSV import, the limit is 10,000 lines per transaction.

Note: As of the 2016.1 WSDL, another sublist, the accountingBookDetailList, is also exposed to web services. This sublist is relevant only if you use the Multi-Book Accounting feature. For details on this sublist, see the SuiteTalk Schema Browser's accountingBookDetailList reference page.

Intercompany Journal Entry Code Samples

For examples of how to structure your journal entry code, refer to the following sections:

- Adding an Intercompany Journal Entry
- Adding a Book Specific Intercompany Journal Entry

Adding an Intercompany Journal Entry

The following example shows how to create an intercompany journal entry record that has four sublist lines.

C#

```
private void addInterCompanyJournalEntry()
{

    InterCompanyJournalEntry newInterCompanyJournalEntry = new InterCompanyJournalEntry();
      newInterCompanyJournalEntry.externalId = "11890-A";

      RecordRef mySubsidiary = new RecordRef();
      mySubsidiary.internalId = "1";
      newInterCompanyJournalEntry.subsidiary = mySubsidiary;

      RecordRef myToSubsidiary = new RecordRef();
      myToSubsidiary.internalId = "10";
      newInterCompanyJournalEntry.toSubsidiary = myToSubsidiary;

      RecordRef myCurrency = new RecordRef();
      myCurrency.internalId = "1";
      newInterCompanyJournalEntry.currency = myCurrency;

      RecordRef myDebitAccount = new RecordRef();
      myDebitAccount.internalId = "1";

    RecordRef myCreditAccount = new RecordRef();
      myCreditAccount.internalId = "3";

      RecordRef mySecondDebitAccount = new RecordRef();
      mySecondDebitAccount.internalId = "214";

      RecordRef mySecondCreditAccount = new RecordRef();
      mySecondCreditAccount.internalId = "215";

      newInterCompanyJournalEntry.lineList = new InterCompanyJournalEntryLineList();
      newInterCompanyJournalEntry.lineList.line = new InterCompanyJournalEntryLine[4];

      newInterCompanyJournalEntry.lineList.line[0] = new InterCompanyJournalEntryLine();
      newInterCompanyJournalEntry.lineList.line[0].lineSubsidiary = mySubsidiary;
      newInterCompanyJournalEntry.lineList.line[0].account = myDebitAccount;
      newInterCompanyJournalEntry.lineList.line[0].debit = 920000;
      newInterCompanyJournalEntry.lineList.line[0].debitSpecified = true;

      newInterCompanyJournalEntry.lineList.line[1] = new InterCompanyJournalEntryLine();
      newInterCompanyJournalEntry.lineList.line[1].lineSubsidiary = mySubsidiary;
      newInterCompanyJournalEntry.lineList.line[1].account = myCreditAccount;
      newInterCompanyJournalEntry.lineList.line[1].credit = 920000;
      newInterCompanyJournalEntry.lineList.line[1].creditSpecified = true;

      newInterCompanyJournalEntry.lineList.line[2] = new InterCompanyJournalEntryLine();
      newInterCompanyJournalEntry.lineList.line[2].lineSubsidiary = myToSubsidiary;
      newInterCompanyJournalEntry.lineList.line[2].account = mySecondDebitAccount;
      newInterCompanyJournalEntry.lineList.line[2].debit = 920000;
      newInterCompanyJournalEntry.lineList.line[2].debitSpecified = true;

      newInterCompanyJournalEntry.lineList.line[3] = new InterCompanyJournalEntryLine();
      newInterCompanyJournalEntry.lineList.line[3].lineSubsidiary = myToSubsidiary;
      newInterCompanyJournalEntry.lineList.line[3].account = mySecondCreditAccount;
      newInterCompanyJournalEntry.lineList.line[3].credit = 920000;
      newInterCompanyJournalEntry.lineList.line[3].creditSpecified = true;

      _service.add(newInterCompanyJournalEntry);

}
```

SOAP Request

```
<soap:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record externalId="11890-A" xsi:type="q1:InterCompanyJournalEntry" xmlns:q1="urn:general
_2017_1.transactions.webservices.netsuite.com">
         <q1:currency internalId="1"/>
         <q1:subsidiary internalId="1"/>
         <q1:toSubsidiary internalId="10"/>
         <q1:lineList>
            <q1:line>
               <q1:lineSubsidiary internalId="1"/>
               <q1:account internalId="1"/>
               <q1:debit>920000</q1:debit>
            </q1:line>
            <q1:line>
               <q1:lineSubsidiary internalId="1"/>
               <q1:account internalId="3"/>
               <q1:credit>920000</q1:credit>
            </q1:line>
            <q1:line>
```

```
                <q1:lineSubsidiary internalId="10"/>
                <q1:account internalId="214"/>
                <q1:debit>920000</q1:debit>
             </q1:line>
             <q1:line>
                <q1:lineSubsidiary internalId="10"/>
                <q1:account internalId="215"/>
                <q1:credit>920000</q1:credit>
             </q1:line>
          </q1:lineList>
       </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="interCompanyJournalEntry" externalId=
"11890-A" internalId="522" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.co
m"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Adding a Book Specific Intercompany Journal Entry

The following example shows how to create a book specific intercompany journal entry record that has four sublist lines.

C#

```
private void addBookSpecificInterCompanyJournalEntry()
{

   InterCompanyJournalEntry newBookSpecificInterCompanyJournalEntry = new InterCompanyJournalEn
try();
   newBookSpecificInterCompanyJournalEntry.externalId = "11895-A";

   RecordRef mySubsidiary = new RecordRef();
   mySubsidiary.internalId = "1";
   newBookSpecificInterCompanyJournalEntry.subsidiary = mySubsidiary;

   RecordRef myToSubsidiary = new RecordRef();
   myToSubsidiary.internalId = "10";
   newBookSpecificInterCompanyJournalEntry.toSubsidiary = myToSubsidiary;

   RecordRef myAccountingBook = new RecordRef();
   myAccountingBook.internalId = "2";
   newBookSpecificInterCompanyJournalEntry.accountingBook = myAccountingBook;
   RecordRef myDebitAccount = new RecordRef();
   myDebitAccount.internalId = "1";

   RecordRef myCreditAccount = new RecordRef();
   myCreditAccount.internalId = "3";

   RecordRef mySecondDebitAccount = new RecordRef();
   mySecondDebitAccount.internalId = "214";

   RecordRef mySecondCreditAccount = new RecordRef();
   mySecondCreditAccount.internalId = "215";

   newBookSpecificInterCompanyJournalEntry.lineList = new InterCompanyJournalEntryLineList();
   newBookSpecificInterCompanyJournalEntry.lineList.line = new InterCompanyJournalEntryLine[4];


   newBookSpecificInterCompanyJournalEntry.lineList.line[0] = new InterCompanyJournalEntryLine(
);
   newBookSpecificInterCompanyJournalEntry.lineList.line[0].lineSubsidiary = mySubsidiary;
   newBookSpecificInterCompanyJournalEntry.lineList.line[0].account = myDebitAccount;
   newBookSpecificInterCompanyJournalEntry.lineList.line[0].debit = 45000;
   newBookSpecificInterCompanyJournalEntry.lineList.line[0].debitSpecified = true;

   newBookSpecificInterCompanyJournalEntry.lineList.line[1] = new InterCompanyJournalEntryLine(
);
   newBookSpecificInterCompanyJournalEntry.lineList.line[1].lineSubsidiary = mySubsidiary;
   newBookSpecificInterCompanyJournalEntry.lineList.line[1].account = myCreditAccount;
   newBookSpecificInterCompanyJournalEntry.lineList.line[1].credit = 45000;
   newBookSpecificInterCompanyJournalEntry.lineList.line[1].creditSpecified = true;

   newBookSpecificInterCompanyJournalEntry.lineList.line[2] = new InterCompanyJournalEntryLine(
);
   newBookSpecificInterCompanyJournalEntry.lineList.line[2].lineSubsidiary = myToSubsidiary;
   newBookSpecificInterCompanyJournalEntry.lineList.line[2].account = mySecondDebitAccount;
   newBookSpecificInterCompanyJournalEntry.lineList.line[2].debit = 45000;
   newBookSpecificInterCompanyJournalEntry.lineList.line[2].debitSpecified = true;

  newBookSpecificInterCompanyJournalEntry.lineList.line[3] = new InterCompanyJournalEntryLine();

   newBookSpecificInterCompanyJournalEntry.lineList.line[3].lineSubsidiary = myToSubsidiary;
   newBookSpecificInterCompanyJournalEntry.lineList.line[3].account = mySecondCreditAccount;
   newBookSpecificInterCompanyJournalEntry.lineList.line[3].credit = 45000;
   newBookSpecificInterCompanyJournalEntry.lineList.line[3].creditSpecified = true;

   _service.add(newBookSpecificInterCompanyJournalEntry);

}
```

SOAP Request

```
<soap:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record externalId="11895-A" xsi:type="q1:InterCompanyJournalEntry" xmlns:q1="urn:general
```

```
_2017_1.transactions.webservices.netsuite.com">
         <q1:accountingBook internalId="2"/>
         <q1:subsidiary internalId="1"/>
         <q1:toSubsidiary internalId="10"/>
         <q1:lineList>
            <q1:line>
               <q1:lineSubsidiary internalId="1"/>
               <q1:account internalId="1"/>
               <q1:debit>45000</q1:debit>
            <q1:line>
            <q1:line>
               <q1:lineSubsidiary internalId="1"/>
               <q1:account internalId="3"/>
               <q1:credit>45000</q1:credit>
            </q1:line>
            <q1:line>
               <q1:lineSubsidiary internalId="10"/>
               <q1:account internalId="214"/>
               <q1:debit>45000</q1:debit>
            </q1:line>
            <q1:line>
               <q1:lineSubsidiary internalId="10"/>
               <q1:account internalId="215"/>
               <q1:credit>45000</q1:credit>
            </q1:line>
         </q1:lineList>
      </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="interCompanyJournalEntry" externalId=
"11895-A" internalId="621" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.co
m"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Common Errors With Intercompany Journal Entries

When you are working with intercompany journal entry records, you may see following failure messages in your SOAP responses.

This record already exists

This error indicates indicates that you attempted to add an intercompany journal entry using an external ID that is already in use.

Invalid account reference key x for linesubsidiary y

This error signifies a problem with one of your lines. Specifically, it indicates that you used an account that is not valid for the subsidiary identified by the lineSubsidiary field.

Invalid account reference key x for linesubsidiary null

This error signifies a problem with one of your lines. Specifically, it indicates that you failed to include a value for the lineSubsidiary field.

Intercompany Transfer Order

In accounts using NetSuite OneWorld, the Intercompany Transfer Order transaction is used to move inventory from a location for one subsidiary to a location for another subsidiary. The intercompany transfer order record is defined in the tranInvt (inventory) XSD.

For information about how intercompany transfer orders are used, see the help topic Intercompany Inventory Transfers - Non-Arm's Length.

Supported Operations

The following operations can be used to modify intercompany transfer order records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's intercompany transfer order reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

It is recommended that you read the usage notes for Transfer Order, because these notes also apply to the Intercompany Transfer Order. The following additional notes apply.

Source Subsidiary and Destination Subsidiary

An intercompany transfer order must have a Subsidiary (meaning the source subsidiary) and a Destination Subsidiary value.

All transfer order line items must be available to both the source and destination subsidiaries.

The Source Location for a transfer order must be from the (source) Subsidiary and the Destination Location must be from the Destination Subsidiary.

The Department and Class values for a transfer order must be from the (source) Subsidiary. These values are used for shipment transactions.

You must have permissions for both the source (Subsidiary) and Destination Subsidiary to create or edit an intercompany transfer order. Permissions for both the Source Location and Destination location also are required, whether the transfer order status is set to Pending Approval or Pending Fulfillment.

Transfer Pricing Notes and the Use Item Cost as Transfer Cost Preference

Intercompany transfer orders can be entered when the Use Item Cost as Transfer Cost preference is enabled on the Order Management subtab at Setup > Accounting >Accounting Preferences. Whether the preference is enabled or disabled, an intercompany transfer order uses the transfer price instead of the item cost for costing calculations.

The transfer order Transfer Price value defaults to the Transfer Price listed on the item record, but can be changed when the transfer order is created or edited.

- The item record Transfer Price is in the currency of the root parent subsidiary. On the transfer order, the Transfer Price should be converted to the correct value in the currency of the source subsidiary.
- If the item record Transfer Price is not set, the Transfer Price on the transfer order is 0. In this case, be sure to edit the transfer order Transfer Price as necessary.

The difference between the item cost and the transfer order Transfer Price is recorded in a Gain/Loss account, and G/L impact occurs after item receipt, as shown in the following table:

Account	Debit	Credit	Subsidiary
Inventory Asset	X		Destination Subsidiary
Inventory in Transit		X	Source Subsidiary
Intercompany Payable/Receivable Intercompany Clearing		X	Destination Subsidiary (Payable)
Intercompany Payable/Receivable Intercompany Clearing	X		Source Subsidiary (Receivable)

Important: When NetSuite released the Intercompany Time and Expenses feature, the first set of accounts created were named Intercompany Payable/Receivable XXX, where XXX denoted the currency ISO code. In Version 2013 Release 1 NetSuite OneWorld introduced the Intercompany Clearing XXX account. This new account replaced the Intercompany Payable/Receivable Account for new accounts because the existing accounts were being used by the Intercompany Elimination feature. The change applied to only new accounts. Existing accounts were not renamed. In Version 2014 Release 1, NetSuite OneWorld introduced new intercompany clearing accounts for payable and receivable that are not currency locked. These new clearing accounts are used for intercompany transactions. All existing currency-locked intercompany clearing accounts (the Intercompany Payable/Receivable accounts) are now child accounts of the new clearing account. For more information, see the help topics Enabling Intercompany Time and Expenses and Intercompany Elimination Overview.

Be aware that the exchange rate between subsidiaries' currencies can affect the G/L impact to each subsidiary.

Inventory Adjustment

The inventory adjustment transaction changes the quantity and value of an inventory item without entering a purchase order. For example, this transaction can be used to account for clerical errors, changes in cost, thefts, or miscounts. If you use the LIFO or FIFO costing methods, this transaction preserves the quantity and value of an inventory item and preserves the costing history of the item.

Important: NetSuite recommends that you do not delete or change inventory transactions dated prior to an inventory distribution, as this can cause difficulties maintaining accurate inventory data.

For more details about this type of transaction, see the help topic Adjusting Inventory.

The inventory adjustment record is defined in the tranInvt (inventory) XSD.

Supported Operations

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's inventory adjustment reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the Inventory Detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Inventory Cost Revaluation

The inventory cost revaluation record is used to recalculate the value of items configured to use standard costing.

This record is defined in the tranInvt (inventory) XSD, where it is called *InventoryCostRevaluation*.

This record is available when the Standard Costing feature is enabled at Setup > Company > Enable Features, on the Items & Inventory subtab. When the feature is enabled, you can access the inventory cost revaluation record in the UI by choosing Transactions > Inventory > Revalue Inventory Cost.

Supported Operations

add | addList | delete | deleteList | get | getList | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's inventory cost revaluation reference page. For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Code Samples

The following sample shows how to create and update inventory cost revaluation records.

```
public void inventoryCostRevaluationSample() throws Exception
{
   c = new NLWsClient(new String[]{“address”,"psswrd", "1234321", "3"}, 5068, “link” // protect
ed NLWsClient c = null;
   c.useRequestLevelCredentials();

   String icrId =  null;
   try
   {
      // Create record
      InventoryCostRevaluation icr = new InventoryCostRevaluation();
      icr.setSubsidiary(createRR("1"));
      icr.setItem(createRR("1"));
      icr.setAccount(createRR("1"));
      icr.setLocation(createRR("1"));

      // Cost component sublist
      InventoryCostRevaluationCostComponent costComponentLine1 = new InventoryCostRevaluationCo
stComponent();
      costComponentLine1.setCost(2.0);
      costComponentLine1.setQuantity(1.0);
      costComponentLine1.setComponentItem(createRR("1")); // Component Item Id
      icr.setCostComponentList(new InventoryCostRevaluationCostComponentList(new InventoryCostR
evaluationCostComponent[]{costComponentLine1},false));

      // Add record
      icrId = c.addRecord(icr);

      // Update
      InventoryCostRevaluation icrUpdated = new InventoryCostRevaluation();
      icrUpdated.setInternalId(icrId);
      icrUpdated.setMemo("Some memo");

      c.updateRecord(icrUpdated);

      // Load record
      c.getRecord(icrId, RecordType.inventoryCostRevaluation);

   }
   finally
   {
      if(icrId==null){
      // Delete
      c.deleteRecord(new RecordRef(null,icrId,null,RecordType.inventoryCostRevaluation));
      }
   }
}
```

Inventory Transfer

The Inventory Transfer transaction posts details about per-location item inventory level changes when items are transferred between two locations. This basic inventory transfer decreases items in the source location and increases them in the receiving location, all in one step.

This transaction is available when the Locations feature and the Multi-Location Inventory (MLI) feature are enabled. For more details, see the help topic Transferring Inventory.

The inventory transfer record is defined in the tranInvt (inventory) XSD.

Note: The transfer order is a more complex transaction that can be used to schedule and track the individual steps of the inventory transfer process through each stage of a transfer process in which items are moved from one location to another over a period of time. A transfer order, unlike an inventory transfer, can go through an approval process. For details about the Transfer Order record, see the help topic Inventory Transfer Orders.

Supported Operations

The following operations can be used with inventory transfer records:

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's inventory transfer reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the inventory detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Invoice

An invoice transaction creates a bill for goods and/or services sold to a customer for which payment is not received at the time of delivery. The invoice indicates terms of payment that specify payment is to be received. Invoice aging tracks how long the amount has been due as payable. Each invoice consists of multiple line items whose sales amounts add up to the total of the invoice. Companies that bill costs back to customers can identify billable costs, mark them up and add them to invoices.

For more details about this type of transaction, see the help topic Custom Workflow Based Invoice Approval.

The invoice record is defined in the tranSales (sales) XSD.

Supported Operations

The following operations can be used with the invoice record:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's invoice reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Initializing Invoices

You can initialize an invoice from a Customer, an Estimate/Quote, an Opportunity, or a Sales Order.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

Working with Invoice Sublists

The SuiteTalk Schema Browser includes all sublists associated with the Invoice record. See the following information for usage notes regarding specific Invoice sublists. Usage notes are not provided for every sublist type.

- InvoiceSalesTeamList
- InvoiceItemCostList
- InvoiceExpCostList
- InvoiceTimeList
- GiftCertRedemptionList

InvoiceSalesTeamList

The InvoiceSalesTeam list defines the sales team for a specific invoice. This list is only available when the Team Selling feature is enabled in the account.

On add, a single line item is entered by default when a sales rep is currently associated with the entity set in the entity field. When a sales group is currently associated with the entity, the list is populated by the group.

InvoiceItemCostList

These fields map to the Billable Items subtab on the Item list of an invoice record. This is where you define how and when to bill item costs back to a customer. Billing items back to customers enables you to purchase items and supplies for an order or job, and then bill the cost to the customer. The Bill Costs To Customers feature must be enabled to use this list.

InvoiceExpCostList

These fields map to the Billable Expenses subtab on the Item list of an invoice record. This is where you define how and when to bill expenses back to a customer. The Bill Costs To Customers feature must be enabled to use this list.

InvoiceTimeList

These fields map to the Billable Time subtab on the Item list of an invoice record. This is where you define how and when to bill time back to a customer. The Bill Costs To Customers feature must be enabled to use this list.

GiftCertRedemptionList

This sublist is available on the invoice, sales order, and cash sale records.

Associating Invoices with Opportunities

You can associate estimates, cash sales, sales orders, and invoices with opportunities. After a transaction other than an estimate is associated with an opportunity, the opportunity's status is automatically set to **Closed Won**. After an opportunity's status is set to **Closed Won**, it is no longer available to be selected on other cash sale, sales order, or invoice records.

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the inventory detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Item Demand Plan

The item demand plan record is available for inventory items when the Demand Planning feature is enabled, and for assembly/BOM items when both the Demand Planning and Work Orders features are enabled. Item demand plans can be added for items where the **supplyReplenishMethod** field is set to Time Phased.

An item demand plan transaction stores the quantity expected to be needed, during specified time periods, for an item. NetSuite supports three types of demand plans: monthly, weekly, and daily.

For more information about using item demand plans, see the help topic Demand Planning.

Each demand plan includes values for body fields used to uniquely identify the record, such as item, startDate, endDate, location (when Multi-Location Inventory is enabled), and subsidiary (for NetSuite OneWorld). A demandPlanCalendarType body field indicates the time period used in the demand plan: monthly, weekly, or daily. The following identifier fields are supported: externalId, internalId, and demandPlanId.

Demand plan sublist data is represented by a periodDemandPlan matrix list. This data varies according to the demand plan type.

- For a monthly plan, each sublist instance represents a month and requires a startDate and a quantity.
- For a weekly plan, each sublist instance represents a week and requires a startDate and a quantity.
- For a daily plan, each sublist instance also represents a week, and requires a startDate, at least one quantity, and at least one dayOfTheWeek. Note that it can have up to 7 values for quantity and for dayOfTheWeek.

The item demand plan record is defined in the transactions demandPlanning XSD.

Supported Operations

add | addList | delete | deleteList | get | | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s item demand plan reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Understanding Start Date and End Date Fields

Each item demand plan record includes body-level startDate and endDate fields and additional startDate and endDate fields for each sublist line.

Body-Level Dates

Body-level startDate and endDate fields are required. The body-level startDate field defaults to January 1 of the current year, and the body-level endDate field defaults to December 31 of the current year, if values are not provided.

Important: Edits are only supported for records where the body-level startDate and endDate fields are within three years of the current year.

Line-Level Dates

Line-level startDate and endDate fields must fall within body-level startDate and endDate fields.

Line-level endDate fields are read-only.

For monthly demand plans, every line-level startDate value must correspond to the first day of a month. For weekly and daily demand plans, every line-level startDate value must correspond to the first day of a week, based on the company preference for First Day of Week set at Setup > Company > General Preferences. (The default is Sunday.)

Demand Planning Sublist

Updates to the periodDemandPlan sublist respect the replaceAll attribute. The handling of replaceAll for matrix sublists like periodDemandPlan is more involved than its handling for most keyed sublists. For details, see the help topic Matrix Sublists and replaceAll.

The startDate field is required and is the key for this sublist.

A get operation only returns lines that have quantity values.

Searching Demand Plans

An advanced search (getSavedSearch) of demand plans returns lines. A basic search returns the whole record.

The only supported filters are the body-level startDate and endDate fields. These filters are required.

Sample Code

The following code illustrates how to create monthly, weekly, and daily item demand plans.

Monthly Item Demand Plan Java

```
public void testAddDemandPlanningMonthly() throws Exception
                {
                                c.setCredentials(CRED_DEV_DB93);
                                c.useRequestLevelCredentials();
                                ItemDemandPlan dp = new ItemDemandPlan();

                                // subsidiary
                                RecordRef subsidiary = new RecordRef();
                                subsidiary.setInternalId("1");
                                subsidiary.setType(RecordType.subsidiary);
                                dp.setSubsidiary(subsidiary);
                                dp.setExternalId("1234567");
                                // location
                                RecordRef location = new RecordRef();
                                location.setInternalId("7"); // Location US 2
                                location.setType(RecordType.location);
                                dp.setLocation(location);

                                // inventory item
                                RecordRef inventoryItem = new RecordRef();
                                inventoryItem.setInternalId("202"); // Inventory Item with "Tim
e Phased" Replenishment Method
                                inventoryItem.setType(RecordType.inventoryItem);
                                dp.setItem(inventoryItem);

                                dp.setStartDate(new GregorianCalendar(2011,0,1));
```

```
                        dp.setDemandPlanCalendarType(DemandPlanCalendarType._monthly);

                        // demand plan matrix
                        DemandPlanMatrix matrix = new DemandPlanMatrix();
                        DemandPlan[] demandPlans = new DemandPlan[2];
                        demandPlans[0] = new DemandPlan();
                        demandPlans[1] = new DemandPlan();

                        // 1/1/2011
                        demandPlans[0].setStartDate(new GregorianCalendar(2011,0,1));
                        PeriodDemandPlanList planList = new PeriodDemandPlanList();
                        PeriodDemandPlan[] plans = new PeriodDemandPlan[1];
                        plans[0] = new PeriodDemandPlan();

                        plans[0].setQuantity(5.0);
                        planList.setPeriodDemandPlan(plans);
                        demandPlans[0].setPeriodDemandPlanList(planList);

                        // 2/1/2011
                        demandPlans[1].setStartDate(new GregorianCalendar(2011,1,1));
                        PeriodDemandPlanList planList2 = new PeriodDemandPlanList();
                        PeriodDemandPlan[] plans2 = new PeriodDemandPlan[1];
                        plans2[0] = new PeriodDemandPlan();

                        plans2[0].setQuantity(7.0);
                        planList2.setPeriodDemandPlan(plans2);
                        demandPlans[1].setPeriodDemandPlanList(planList2);
                        matrix.setDemandPlan(demandPlans);
                        dp.setDemandPlanMatrix(matrix);
                        c.addRecord(dp);
        }
```

Monthly Item Demand Plan SOAP Request

```
 <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
           <record xsi:type="ns6:ItemDemandPlan" xmlns:ns6="urn:demandplanning_2017_1.transact
ions.webservices.netsuite.com">
              <ns6:subsidiary internalId="1" type="subsidiary" xsi:type="ns7:RecordRef" xmlns:
ns7="urn:core_2017_1.platform.webservices.netsuite.com"/>
              <ns6:location internalId="7" type="location" xsi:type="ns8:RecordRef" xmlns:ns8=
"urn:core_2017_1.platform.webservices.netsuite.com"/>
              <ns6:item internalId="199" type="inventoryItem" xsi:type="ns9:RecordRef" xmlns:n
s9="urn:core_2017_1.platform.webservices.netsuite.com"/>
              <ns6:startDate xsi:type="xsd:dateTime">2009-01-01T08:00:00.000Z</ns6:startDate>

              <ns6:endDate xsi:type="xsd:dateTime">2011-12-31T08:00:00.000Z</ns6:endDate>
              <ns6:demandPlanCalendarType xsi:type="ns10:DemandPlanCalendarType" xmlns:ns10="u
rn:types.demandplanning_2017_1.transactions.webservices.netsuite.com">_monthly</ns6:demandPlanC
alendarType>
              <ns6:demandPlanMatrix xsi:type="ns6:DemandPlanMatrix">
                 <ns6:demandPlan xsi:type="ns6:DemandPlan">
                    <ns6:startDate xsi:type="xsd:dateTime">2011-11-20T16:00:00.000Z</ns6:start
Date>
```

```
                    <ns6:periodDemandPlanList xsi:type="ns6:PeriodDemandPlanList">
                       <ns6:periodDemandPlan xsi:type="ns6:PeriodDemandPlan">
                          <ns6:quantity xsi:type="xsd:double">68.0</ns6:quantity>
                       </ns6:periodDemandPlan>
                    </ns6:periodDemandPlanList>
                 </ns6:demandPlan>
              </ns6:demandPlanMatrix>
           </record>
        </add>
```

Weekly Item Demand Plan Java

```
public void testAddDemandPlanningWeekly() throws Exception
                {
                                c.setCredentials(CRED_DEV_DB93);
                                c.useRequestLevelCredentials();
                                ItemDemandPlan dp = new ItemDemandPlan();

                                // subsidiary
                                RecordRef subsidiary = new RecordRef();
                                subsidiary.setInternalId("1");
                                subsidiary.setType(RecordType.subsidiary);
                                dp.setSubsidiary(subsidiary);
                                dp.setExternalId("2345678");
                                // location
                                RecordRef location = new RecordRef();
                                location.setInternalId("7"); // Location US 2
                                location.setType(RecordType.location);
                                dp.setLocation(location);

                                // inventory item
                                RecordRef inventoryItem = new RecordRef();
                                inventoryItem.setInternalId("202"); // Inventory Item with "Tim
e Phased" Replenishment Method
                                inventoryItem.setType(RecordType.inventoryItem);
                                dp.setItem(inventoryItem);

                                dp.setStartDate(new GregorianCalendar(2011,0,1));
                                dp.setDemandPlanCalendarType(DemandPlanCalendarType._weekly);

                                // demand plan matrix
                                DemandPlanMatrix matrix = new DemandPlanMatrix();
                                DemandPlan[] demandPlans = new DemandPlan[2];
                                demandPlans[0] = new DemandPlan();
                                demandPlans[1] = new DemandPlan();

                                // 1/2/2011
                                demandPlans[0].setStartDate(new GregorianCalendar(2011,0,2));
                                PeriodDemandPlanList planList = new PeriodDemandPlanList();
                                PeriodDemandPlan[] plans = new PeriodDemandPlan[1];
                                plans[0] = new PeriodDemandPlan();

                                plans[0].setQuantity(5.0);
                                planList.setPeriodDemandPlan(plans);
                                demandPlans[0].setPeriodDemandPlanList(planList);

                                // 1/9/2011
                                demandPlans[1].setStartDate(new GregorianCalendar(2011,0,9));
                                PeriodDemandPlanList planList2 = new PeriodDemandPlanList();
                                PeriodDemandPlan[] plans2 = new PeriodDemandPlan[1];
                                plans2[0] = new PeriodDemandPlan();

                                plans2[0].setQuantity(7.0);
                                planList2.setPeriodDemandPlan(plans2);
                                demandPlans[1].setPeriodDemandPlanList(planList2);
                                matrix.setDemandPlan(demandPlans);
                                dp.setDemandPlanMatrix(matrix);
                                c.addRecord(dp);
                }
```

Weekly Item Demand Plan SOAP Request

```
 <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">


           <record xsi:type="ns6:ItemDemandPlan" xmlns:ns6="urn:demandplanning_2017_1.transact
ions.webservices.netsuite.com">


               <ns6:subsidiary internalId="1" type="subsidiary" xsi:type="ns7:RecordRef" xmlns:
ns7="urn:core_2017_1.platform.webservices.netsuite.com"/>


               <ns6:location internalId="7" type="location" xsi:type="ns8:RecordRef" xmlns:ns8=
"urn:core_2017_1.platform.webservices.netsuite.com"/>


               <ns6:item internalId="199" type="inventoryItem" xsi:type="ns9:RecordRef" xmlns:n
s9="urn:core_2017_1.platform.webservices.netsuite.com"/>


               <ns6:startDate xsi:type="xsd:dateTime">2011-11-26T02:27:40.242Z</ns6:startDate>


               <ns6:endDate xsi:type="xsd:dateTime">2011-12-24T02:27:40.242Z</ns6:endDate>


               <ns6:demandPlanCalendarType xsi:type="ns10:DemandPlanCalendarType" xmlns:ns10="u
rn:types.demandplanning_2017_1.transactions.webservices.netsuite.com">_weekly</ns6:demandPlanCa
lendarType>


               <ns6:demandPlanMatrix xsi:type="ns6:DemandPlanMatrix">
```

```
                   <ns6:demandPlan xsi:type="ns6:DemandPlan">



                      <ns6:startDate xsi:type="xsd:dateTime">2011-11-20T16:00:00.000Z</ns6:start
Date>


                      <ns6:periodDemandPlanList xsi:type="ns6:PeriodDemandPlanList">




                         <ns6:periodDemandPlan xsi:type="ns6:PeriodDemandPlan">
```

Daily Item Demand Plan Java

```
 public void testAddDemandPlanningDaily() throws Exception
                {
                                c.setCredentials(CRED_DEV_DB93);
                                c.useRequestLevelCredentials();
                                ItemDemandPlan dp = new ItemDemandPlan();

                                // subsidiary
                                RecordRef subsidiary = new RecordRef();
                                subsidiary.setInternalId("1");
                                subsidiary.setType(RecordType.subsidiary);
                                dp.setSubsidiary(subsidiary);

                                // location
                                RecordRef location = new RecordRef();
                                location.setInternalId("7"); // Location US 2
                                location.setType(RecordType.location);
                                dp.setLocation(location);

                                // inventory item
                                RecordRef inventoryItem = new RecordRef();
                                inventoryItem.setInternalId("199"); // Inventory Item with "Tim
e Phased" Replenishment Method
                                inventoryItem.setType(RecordType.inventoryItem);
                                dp.setItem(inventoryItem);

                                dp.setStartDate(new GregorianCalendar(2009,0,1));
                                dp.setEndDate(new GregorianCalendar(2011,11,31));
                                dp.setDemandPlanCalendarType(DemandPlanCalendarType._daily);

                                // demand plan matrix
                                DemandPlanMatrix matrix = new DemandPlanMatrix();
                                DemandPlan[] demandPlans = new DemandPlan[1];
                                demandPlans[0] = new DemandPlan();
                                demandPlans[0].setStartDate(new GregorianCalendar(2011,0,2));

                                PeriodDemandPlanList planList = new PeriodDemandPlanList();

                                PeriodDemandPlan[] plans = new PeriodDemandPlan[2];
                                plans[0] = new PeriodDemandPlan();
                                plans[0].setDayOfTheWeek(DayOfTheWeek._sunday);
                                plans[0].setQuantity(5.0);

                                plans[1] = new PeriodDemandPlan();
                                plans[1].setDayOfTheWeek(DayOfTheWeek._monday);
                                plans[1].setQuantity(6.0);
                                planList.setPeriodDemandPlan(plans);
                                demandPlans[0].setPeriodDemandPlanList(planList);
                                matrix.setDemandPlan(demandPlans);
                                dp.setDemandPlanMatrix(matrix);
                                c.addRecord(dp);
                }
```

Daily Item Demand Plan SOAP Request

```
 <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record xsi:type="ns6:ItemDemandPlan" xmlns:ns6="urn:demandplanning_2017_1.transact
ions.webservices.netsuite.com">
               <ns6:subsidiary internalId="1" type="subsidiary" xsi:type="ns7:RecordRef" xmlns:
ns7="urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns6:location internalId="7" type="location" xsi:type="ns8:RecordRef" xmlns:ns8=
"urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns6:item internalId="199" type="inventoryItem" xsi:type="ns9:RecordRef" xmlns:n
s9="urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns6:startDate xsi:type="xsd:dateTime">2009-01-01T08:00:00.000Z</ns6:startDate>

               <ns6:endDate xsi:type="xsd:dateTime">2011-12-31T08:00:00.000Z</ns6:endDate>
               <ns6:demandPlanCalendarType xsi:type="ns10:DemandPlanCalendarType" xmlns:ns10="u
rn:types.demandplanning_2017_1.transactions.webservices.netsuite.com">_daily</ns6:demandPlanCal
endarType>
               <ns6:demandPlanMatrix xsi:type="ns6:DemandPlanMatrix">
                  <ns6:demandPlan xsi:type="ns6:DemandPlan">
                     <ns6:startDate xsi:type="xsd:dateTime">2011-01-02T08:00:00.000Z</ns6:start
Date>
                     <ns6:periodDemandPlanList xsi:type="ns6:PeriodDemandPlanList">
                        <ns6:periodDemandPlan xsi:type="ns6:PeriodDemandPlan">
                           <ns6:quantity xsi:type="xsd:double">5.0</ns6:quantity>
                           <ns6:dayOfTheWeek xsi:type="ns11:DayOfTheWeek" xmlns:ns11="urn:types
.demandplanning_2017_1.transactions.webservices.netsuite.com">_sunday</ns6:dayOfTheWeek>

                        </ns6:periodDemandPlan>
                        <ns6:periodDemandPlan xsi:type="ns6:PeriodDemandPlan">
                           <ns6:quantity xsi:type="xsd:double">6.0</ns6:quantity>
                           <ns6:dayOfTheWeek xsi:type="ns12:DayOfTheWeek" xmlns:ns12="urn:types
.demandplanning_2017_1.transactions.webservices.netsuite.com">_monday</ns6:dayOfTheWeek>

                        </ns6:periodDemandPlan>
```

```
                    </ns6:periodDemandPlanList>
                </ns6:demandPlan>
            </ns6:demandPlanMatrix>
        </record>
    </add>

                            <ns6:quantity xsi:type="xsd:double">68.0</ns6:quantity>




                        </ns6:periodDemandPlan>




                    </ns6:periodDemandPlanList>


                </ns6:demandPlan>


            </ns6:demandPlanMatrix>


        </record>

    </add>
```

Item Fulfillment

An item fulfillment transaction records the shipment of some or all items on an order to the customer. The processes for item fulfillment transactions depend on whether the Advanced Shipping feature is enabled.

- If Advanced Shipping is not enabled, the fulfillment and invoicing processes are combined. When an item fulfillment is created, a related invoice is created at the same time.
- If Advanced Shipping is enabled, fulfillment and invoicing are two independent processes, and shipments can be recorded separately from billing.

Note: Whether or not Advanced Shipping is enabled, order fulfillments should always be entered against sales orders to track the status of items and orders.

For more details about this type of transaction, see the help topic Order Fulfillment.

The item fulfillment record is defined in the tranSales (sales) XSD.

Supported Operations

The following operations can be used to manipulate the item fulfillment record.

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's item fulfillment reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Initializing Item Fulfillments

You can initialize an item fulfillment from a Sales Order, a Transfer Order, an Intercompany Transfer Order, or a Vendor Return Authorization.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

Item Fulfillment Workflow

When submitting an item fulfillment record, NetSuite can initialize this record on the server with data from the related transaction (such as a sales order) referenced in the **createdFrom** field. Then the item fulfillment record is updated with the itemList provided in your request and that data is validated against the initialized data from the referenced transaction.

Partial Fulfillments

The 2009.2 and earlier endpoints allow you to specify that only some of the line items in a referenced transaction should be fulfilled. The 2010.1 and later endpoints do not support partial fulfillments of line items from a transaction referenced in the createdFrom field. This behavior is consistent with item fulfillments in the UI and in SuiteScript.

Instead of using the createdFrom field, you can use the initialize operation to create an item fulfillment in web services. You can then get the item fulfillment record and specify which lines should be fulfilled before submitting the record.

Important: Validation for Fed Ex or UPS related fields occurs after a record is routed to FedEx or UPS respectively and NOT by the NetSuite application as in the UI. Therefore, it is the responsibility of the client application developer to ensure that the correct fields are populated with the required information for shipping.

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the inventory detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Sample Code

The following code illustrates adding a sales order record with a single item and then fulfilling that order. In this case, the account is set up for shipping with FedEx and FedEx is set as the method of shipping on the sales order record.

First, a sales order is created, set to the pendingFulfillment status, and populated with an itemList.

Create Sales Order Java

```
sessMgr.login();
SalesOrder salesOrder = new SalesOrder();
salesOrder.setEntity(Util.makeRecordRef("87", RecordType.customer));
salesOrder.setOrderStatus(SalesOrderOrderStatus._pendingFulfillment);
SalesOrderItem soi = new SalesOrderItem();
soi.setItem (Util.makeRecordRef("15", RecordType.inventoryItem));
soi.setQuantity(new Double (1));
soi.setAmount(new Double (14));
SalesOrderItemList soil = new SalesOrderItemList(new SalesOrderItem[]{soi}, true);
salesOrder.setItemList(soil);
salesOrder = sessMgr.getNetsuitePort().add(salesOrder);
```

Create Sales Order SOAP Request

```
<soapenv:Body>
<add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<record xsi:type="ns1:SalesOrder" xmlns:ns1="urn:sales_2017_1.transactions.webservices.netsuite
.com">
<ns1:entity internalId="87 customer" xsi:type="ns2:RecordRef" xmlns:ns2="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
<ns1:orderStatus xsi:type="ns3:SalesOrderOrderStatus" xmlns:ns3="urn:types.sales_2017_1.transac
tions.webservices.netsuite.com">_pendingFulfillment</ns1:orderStatus>
<ns1:itemList replaceAll="true" xsi:type="ns1:SalesOrderItemList">
<ns1:item xsi:type="ns1:SalesOrderItem">
<ns1:item internalId="15 inventoryItem" xsi:type="ns4:RecordRef" xmlns:ns4="urn:core_2017_1.pla
tform.webservices.netsuite.com"/>
<ns1:quantity xsi:type="xsd:double">1.0</ns1:quantity>
<ns1:amount xsi:type="xsd:double">14.0</ns1:amount>
</ns1:item>
</ns1:itemList>
</record>
</add>
```

```
</soapenv:Body>
```

Using a Get operation on the sales order created above, the itemList information can be determined.

Get Sales Order SOAP Request

```
<soapenv:Body>
<get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<baseRef internalId="1505 salesOrder" xsi:type="ns1:RecordRef" xmlns:ns1="urn:core_2017_1.platf
orm.webservices.netsuite.com"/>
</get>
</soapenv:Body>
```

Get Sales Order 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="1505" xsi:type="ns2:SalesOrder" xmlns:xsi="http://www.w3.org/2001/XMLSchema
-instance" xmlns:ns2="urn:sales_2017_1.transactions.webservices.netsuite.com">
<ns2:entity internalId="87">
<ns3:name xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">Abe Simpson</ns3:name>
</ns2:entity>
<ns2:tranDate>2006-08-22T00:00:00.000-07:00</ns2:tranDate>
<ns2:tranId>SORD10069</ns2:tranId>
<ns2:orderStatus>_pendingFulfillment</ns2:orderStatus>
<ns2:salesRep internalId="43">
<ns4:name xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com">Jon Baker</ns4:name>
</ns2:salesRep>
<ns2:leadSource internalId="-2">
<ns5:name xmlns:ns5="urn:core_2017_1.platform.webservices.netsuite.com">Ad</ns5:name>
</ns2:leadSource>
<ns2:excludeCommission>false</ns2:excludeCommission>
<ns2:isTaxable>false</ns2:isTaxable>
<ns2:toBePrinted>false</ns2:toBePrinted>
<ns2:toBeEmailed>false</ns2:toBeEmailed>
<ns2:email>asimpson@boo.com</ns2:email>
<ns2:toBeFaxed>false</ns2:toBeFaxed>
<ns2:billAddress>Abe Simpson<br>34 Elm St<br>Great Falls MT</ns2:billAddress>
<ns2:shipAddress>Abe Simpson<br>34 Elm St<br>Great Falls MT</ns2:shipAddress>
<ns2:shipDate>2006-08-22T00:00:00.000-07:00</ns2:shipDate>
<ns2:subTotal>14.0</ns2:subTotal>
<ns2:total>14.0</ns2:total>
<ns2:balance>0.0</ns2:balance>
<ns2:lastModifiedDate>2006-08-22T15:49:00.000-07:00</ns2:lastModifiedDate>
<ns2:status>Pending Fulfillment</ns2:status>
<ns2:itemList>
<ns2:item>
<ns2:item internalId="15">
<ns6:name xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.com">Tongue Depressor</ns6:n
ame>
</ns2:item>
```

```
<ns2:quantity>1.0</ns2:quantity>
<ns2:price internalId="1">
<ns7:name xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">Base Price</ns7:name>
</ns2:price>
<ns2:rate>14.00</ns2:rate>
<ns2:amount>14.0</ns2:amount>
<ns2:commitInventory>_availableQty</ns2:commitInventory>
<ns2:isTaxable>true</ns2:isTaxable>
<ns2:isClosed>false</ns2:isClosed>
<ns2:line>1</ns2:line>
<ns2:quantityBackOrdered>0.0</ns2:quantityBackOrdered>
<ns2:quantityBilled>0.0</ns2:quantityBilled>
<ns2:quantityCommitted>1.0</ns2:quantityCommitted>
<ns2:quantityFulfilled>0.0</ns2:quantityFulfilled>
</ns2:item>
</ns2:itemList>
</record>
</readResponse>
</getResponse>
</soapenv:Body>
```

Item Fulfillment Java

Next the sales order created above is referenced by setting the createdFrom field to the internalId of the sales order. The specific items to be fulfilled are referenced by setting the Line field to the desired item line from the sales order.

```
ItemFulfillment itemFulfillment = new ItemFulfillment();
itemFulfillment.setCreatedFrom(Util.makeRecordRef(salesOrder.getInternalId(), RecordType.salesO
rder));
ItemFulfillmentItemList ifil = new ItemFulfillmentItemList();
ItemFulfillmentItem ifi = new ItemFulfillmentItem();
ifi.setOrderLine(salesOrder.getItemList().getItem(0).getLine());
ifil.setItem(new ItemFulfillmentItem[]{ifi});
itemFulfillment.setItemList(ifil);
itemFulfillment = sessMgr.getNetsuitePort().add(itemFulfillment);
```

Item Fulfillment SOAP Request

```
<soapenv:Body>
<add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<record xsi:type="ns1:ItemFulfillment" xmlns:ns1="urn:sales_2017_1.transactions.webservices.net
suite.com">
<ns1:createdFrom internalId="1505 salesOrder" xsi:type="ns2:RecordRef" xmlns:ns2="urn:core_2017
_1.platform.webservices.netsuite.com"/>
<ns1:itemList replaceAll="false" xsi:type="ns1:ItemFulfillmentItemList">
<ns1:item xsi:type="ns1:ItemFulfillmentItem">
<ns1:orderLine xsi:type="xsd:long">1</ns1:orderLine>
</ns1:item>
</ns1:itemList>
</record>
</add>
</soapenv:Body>
```

Important: Item fulfillment creation is affected by the preference set at Setup > Accounting > Accounting Preferences > Order Management tab > Fulfillment Based on Commitment. The best practice for creating an item fulfillment in web services is to use the initialize operation, as it returns the item fulfillment with all the defaults set and users will not have to specify the item quantities themselves. Following this practice prevents users from generating a partially fulfilled order without knowing it. (See the help topic initialize / initializeList in the NetSuite Help Center for details on this operation.)

Item Receipt

An item receipt transaction records the receipt of returned items from customers. This transaction updates the following information:

- Items on return authorizations are recorded as received.
- Inventory records are updated for the new stock levels.
- Inventory asset accounts are updated with the values of returned items.
- Status of the return is updated.

The item receipt transaction is available when the Advanced Receiving feature is enabled.

For more details about this type of transaction, see the help topics Receiving a Customer Return and Handling Returned Items.

The item receipt record is defined in tranPurch (purchases) XSD.

Supported Operations

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's item receipt reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Initializing Item Receipts

You can initialize an item receipt from a Purchase Order, a Return Authorization, a Transfer Order, or an Intercompany Transfer Order.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

Initializing Item Receipts from Transfer Orders

You can enter partial fulfillments and receipts for transfer orders, and track item costs throughout the transfer process. For details, see the help topic Fulfilling Transfer Orders. To fulfill partial transfer orders, you must have the Use Item Cost as Transfer Cost preference enabled for that transfer order.

When an item receipt is initialized from a transfer order, the InitializeAuxRefType itemFulfillment parameter is returned in request responses. The returned itemFulfillment parameter specifies the item fulfillment record associated with the transfer order.

For information on how to work with the transfer order record in the UI, see the help topic Transferring Inventory. For information on how to work with the record in web services, see Transfer Order.

The following example shows the initialization of an item receipt using the itemFulfillment parameter from a that was fulfilled twice.

Java

```
public void test_item_receipt_from_transfer_order_specify_fulfillment() throws Exception {
ItemReceipt ir = (ItemReceipt) c.initialize(new InitializeRef(null, InitializeRefType.transferO
rder, "308", null), InitializeType.itemReceipt, new InitializeAuxRef(null, InitializeAuxRefType
.itemFulfillment, "309", null))
c.addRecord(ir)
ItemReceipt ir2 = (ItemReceipt) c.initialize(new InitializeRef(null, InitializeRefType.transfer
Order, "308", null), InitializeType.itemReceipt, new InitializeAuxRef(null, InitializeAuxRefTyp
e.itemFulfillment, "310", null))
c.addRecord(ir2)
}
```

SOAP Request

```
<soapenv:Body>
        <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record xsi:type="ns8:ItemReceipt" xmlns:ns8="urn:purchases_2017_1.transactions.web
services.netsuite.com">
                <ns8:createdDate xsi:type="xsd:dateTime">2016-12-06T22:32:00.000Z</ns8:createdD
ate>
                <ns8:lastModifiedDate xsi:type="xsd:dateTime">2016-12-06T22:43:00.000Z</ns8:las
tModifiedDate>
                <ns8:exchangeRate xsi:type="xsd:double">1.0</ns8:exchangeRate>
                <ns8:currencyName xsi:type="xsd:string">USA</ns8:currencyName>
                <ns8:createdFrom xsi:type="ns9:RecordRef" internalId="308" xmlns:ns9="urn:core_
2017_1.platform.webservices.netsuite.com">
                    <ns9:name xsi:type="xsd:string"> #1</ns9:name>
                </ns8:createdFrom>
                <ns8:tranDate xsi:type="xsd:dateTime">2016-12-08T08:00:00.000Z</ns8:tranDate>
                <ns8:postingPeriod xsi:type="ns10:RecordRef" internalId="289" xmlns:ns10="urn:c
ore_2017_1.platform.webservices.netsuite.com">
                    <ns10:name xsi:type="xsd:string">Dec 2016</ns10:name>
                </ns8:postingPeriod>
                <ns8:tranId xsi:type="xsd:string">5</ns8:tranId>
                <ns8:itemFulfillment xsi:type="ns11:RecordRef" internalId="310" xmlns:ns11="urn
:core_2017_1.platform.webservices.netsuite.com">
                    <ns11:name xsi:type="xsd:string">Item Fulfillment #7</ns11:name>
                </ns8:itemFulfillment>
                <ns8:currency xsi:type="ns12:RecordRef" internalId="1" xmlns:ns12="urn:core_201
```

```
7_1.platform.webservices.netsuite.com">
                    <ns12:name xsi:type="xsd:string">USA</ns12:name>
                </ns8:currency>
                <ns8:itemList xsi:type="ns8:ItemReceiptItemList" replaceAll="false">
                    <ns8:item xsi:type="ns8:ItemReceiptItem">
                        <ns8:itemReceive xsi:type="xsd:boolean">true</ns8:itemReceive>
                        <ns8:item xsi:type="ns13:RecordRef" internalId="61" xmlns:ns13="urn:cor
e_2017_1.platform.webservices.netsuite.com">
                            <ns13:name xsi:type="xsd:string">Telephone Headset</ns13:name>
                        </ns8:item>
                        <ns8:orderLine xsi:type="xsd:long">6</ns8:orderLine>
                        <ns8:line xsi:type="xsd:long">6</ns8:line>
                        <ns8:itemName xsi:type="xsd:string">Telephone Headset</ns8:itemName>
                        <ns8:location xsi:type="ns14:RecordRef" internalId="2" xmlns:ns14="urn:
core_2017_1.platform.webservices.netsuite.com">
                            <ns14:name xsi:type="xsd:string">Warehouse - West Coast</ns14:name>

                        </ns8:location>
                        <ns8:onHand xsi:type="xsd:double">0.0</ns8:onHand>
                        <ns8:quantityRemaining xsi:type="xsd:double">48.0</ns8:quantityRemainin
g>
                        <ns8:quantity xsi:type="xsd:double">48.0</ns8:quantity>
                    </ns8:item>
                </ns8:itemList>
            </record>
        </add>
    </soapenv: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.platf
orm.webservices.netsuite.com"/>
                <baseRef xsi:type="platformCore:RecordRef" type="itemReceipt" internalId="409"
xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </writeResponse>
        </addResponse>
    </soapenv:Body>
```

Key Field for ItemReceiptItemList

The orderLine field 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.

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the inventory detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Item Supply Plan

An item supply plan lists the purchase orders or work orders required to ensure that item quantity meets expected demand.

Importing item supply plan data through web services is similar to creating a manual item supply plan in the NetSuite user interface. This type of item supply plan is not derived from any demand plan in the NetSuite system and can be based on data or projections from external sources.

Important: Only one item supply plan per item is supported.

The schema for this record includes:

- Body fields used to identify the record, such as externalId, internalId, item, location (when Multi-Location Inventory is enabled), memo, subsidiary (for NetSuite OneWorld), and units (when Multiple Units of Measure is enabled).

  Note the following:

  - You must provide externalId or iInternal ID values to uniquely identify items, because the Item field may have non-unique values.
  - Values for item, location, and units fields cannot be changed in update operations.
- ItemSupplyPlanOrderList sublist fields that define data for the orders needed to replenish the supply of items.

The item supply plan record is defined in the transactions demandPlanning XSD.

Supported Operations

add | addList | delete | deleteList | get | getDeleted | getList | search | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's item supply plan reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

The item, location, and units body fields cannot be changed in update operations.

An item supply plan's receiptDate cannot be earlier than the orderDate.

The orderCreated field is read-only. It is set to True when an order is generated from an item supply plan.

Code Samples

Each of the following examples creates an item supply plan with two order lines.

SOAP Request

```
/* SOAP request
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record externalId="Grape15763464" xsi:type="ns6:ItemSupplyPlan" xmlns:ns6="urn:dem
andplanning_2017_1.transactions.webservices.netsuite.com">
               <ns6:location internalId="1" type="location" xsi:type="ns7:RecordRef" xmlns:ns7=
"urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns6:item internalId="21" type="inventoryItem" xsi:type="ns8:RecordRef" xmlns:ns
8="urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns6:orderList replaceAll="false" xsi:type="ns6:ItemSupplyPlanOrderList">
                  <ns6:itemSupplyPlanOrder xsi:type="ns6:ItemSupplyPlanOrder">
                     <ns6:orderDate xsi:type="xsd:dateTime">2012-06-11T16:30:04.593Z</ns6:order
Date>
                     <ns6:receiptDate xsi:type="xsd:dateTime">2012-06-11T16:30:04.593Z</ns6:rec
eiptDate>
                     <ns6:quantity xsi:type="xsd:double">3.0</ns6:quantity>
                     <ns6:orderType xsi:type="ns9:ItemSupplyPlanOrderType" xmlns:ns9="urn:types
.demandplanning_2017_1.transactions.webservices.netsuite.com">_purchaseOrder</ns6:orderType>

                  </ns6:itemSupplyPlanOrder>
                  <ns6:itemSupplyPlanOrder xsi:type="ns6:ItemSupplyPlanOrder">
                     <ns6:orderDate xsi:type="xsd:dateTime">2012-07-04T16:30:04.593Z</ns6:order
Date>
                     <ns6:receiptDate xsi:type="xsd:dateTime">2012-07-12T16:30:04.593Z</ns6:rec
eiptDate>
                     <ns6:quantity xsi:type="xsd:double">3.0</ns6:quantity>
                     <ns6:orderType xsi:type="ns10:ItemSupplyPlanOrderType" xmlns:ns10="urn:typ
es.demandplanning_2017_1.transactions.webservices.netsuite.com">_purchaseOrder</ns6:orderType>

                  </ns6:itemSupplyPlanOrder>
               </ns6:orderList>
            </record>
    </add>
   */
```

Java

```
public void addItemSupplyPlan() throws Exception
   {
      this.login(true);

     ItemSupplyPlan isp = new ItemSupplyPlan();

     RecordRef rrSubsidiary = new RecordRef();
     rrSubsidiary.setInternalId("1");
     rrSubsidiary.setType(RecordType.subsidiary);
     isp.setSubsidiary(Util.makeRecordRef("1"));

     RecordRef rrLocation = new RecordRef();
     rrLocation.setInternalId("1");
     rrLocation.setType(RecordType.location);
     isp.setLocation(rrLocation);

     RecordRef rrItem1 = new RecordRef();
     rrItem1.setInternalId("21");
     rrItem1.setType(RecordType.inventoryItem);
     isp.setItem(rrItem1);

     ItemSupplyPlanOrderList ispl = new ItemSupplyPlanOrderList();
     ItemSupplyPlanOrder[] ispo =  new ItemSupplyPlanOrder[2];

     ispo[0] = new ItemSupplyPlanOrder();

     Calendar c = Calendar.getInstance();
     c.set(2012, 5, 12);
     ispo[0].setOrderDate(c);

     Calendar c2 = Calendar.getInstance();
     c2.set(2012, 5, 12);
     ispo[0].setReceiptDate(c2);

     ispo[0].setQuantity(3.0);
     ispo[0].setOrderType(ItemSupplyPlanOrderType._purchaseOrder);

       ispo[1] = new ItemSupplyPlanOrder();

       Calendar c3 = Calendar.getInstance();
     c3.set(2012, 6, 05);
     ispo[1].setOrderDate(c3);

     Calendar c4 = Calendar.getInstance();
     c4.set(2012, 6, 13);
     ispo[1].setReceiptDate(c4);

     ispo[1].setQuantity(3.0);
     ispo[1].setOrderType(ItemSupplyPlanOrderType._purchaseOrder);

       ispl.setItemSupplyPlanOrder(ispo);
       isp.setOrderList(ispl);
      _port.add(isp);
```

```
    }
```

Each of the following examples uses the addList operation.

SOAP Request

```
/*

     <soapenv:Body>
         <addList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record externalId="Grape16979713" xsi:type="ns6:ItemSupplyPlan" xmlns:ns6="urn:dem
andplanning_2017_1.transactions.webservices.netsuite.com">
               <ns6:location internalId="1" type="location" xsi:type="ns7:RecordRef" xmlns:ns7=
"urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns6:item internalId="21" type="inventoryItem" xsi:type="ns8:RecordRef" xmlns:ns
8="urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns6:orderList replaceAll="false" xsi:type="ns6:ItemSupplyPlanOrderList">
                  <ns6:itemSupplyPlanOrder xsi:type="ns6:ItemSupplyPlanOrder">
                     <ns6:orderDate xsi:type="xsd:dateTime">2012-06-11T16:42:59.250Z</ns6:order
Date>
                     <ns6:receiptDate xsi:type="xsd:dateTime">2012-06-11T16:42:59.250Z</ns6:rec
eiptDate>
                     <ns6:quantity xsi:type="xsd:double">3.0</ns6:quantity>
                     <ns6:orderType xsi:type="ns9:ItemSupplyPlanOrderType" xmlns:ns9="urn:types
.demandplanning_2017_1.transactions.webservices.netsuite.com">_purchaseOrder</ns6:orderType>

                  </ns6:itemSupplyPlanOrder>
                  <ns6:itemSupplyPlanOrder xsi:type="ns6:ItemSupplyPlanOrder">
                     <ns6:orderDate xsi:type="xsd:dateTime">2012-07-04T16:42:59.250Z</ns6:order
Date>
                     <ns6:receiptDate xsi:type="xsd:dateTime">2012-07-12T16:42:59.250Z</ns6:rec
eiptDate>
                     <ns6:quantity xsi:type="xsd:double">3.0</ns6:quantity>
                     <ns6:orderType xsi:type="ns10:ItemSupplyPlanOrderType" xmlns:ns10="urn:typ
es.demandplanning_2017_1.transactions.webservices.netsuite.com">_purchaseOrder</ns6:orderType>

                  </ns6:itemSupplyPlanOrder>
               </ns6:orderList>
            </record>
            <record externalId="Strawberry10442165" xsi:type="ns11:ItemSupplyPlan" xmlns:ns11="
urn:demandplanning_2017_1.transactions.webservices.netsuite.com">
               <ns11:location internalId="1" type="location" xsi:type="ns12:RecordRef" xmlns:ns
12="urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns11:item internalId="22" type="inventoryItem" xsi:type="ns13:RecordRef" xmlns:
ns13="urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns11:orderList replaceAll="false" xsi:type="ns11:ItemSupplyPlanOrderList">

                  <ns11:itemSupplyPlanOrder xsi:type="ns11:ItemSupplyPlanOrder">
                     <ns11:orderDate xsi:type="xsd:dateTime">2012-06-11T16:42:59.250Z</ns11:ord
erDate>
                     <ns11:receiptDate xsi:type="xsd:dateTime">2012-06-11T16:42:59.250Z</ns11:r
eceiptDate>
```

```
                    <ns11:quantity xsi:type="xsd:double">3.0</ns11:quantity>
                    <ns11:orderType xsi:type="ns14:ItemSupplyPlanOrderType" xmlns:ns14="urn:ty
pes.demandplanning_2017_1.transactions.webservices.netsuite.com">_purchaseOrder</ns11:orderType
>
                 </ns11:itemSupplyPlanOrder>
                 <ns11:itemSupplyPlanOrder xsi:type="ns11:ItemSupplyPlanOrder">
                    <ns11:orderDate xsi:type="xsd:dateTime">2012-07-04T16:42:59.250Z</ns11:ord
erDate>
                    <ns11:receiptDate xsi:type="xsd:dateTime">2012-07-12T16:42:59.250Z</ns11:r
eceiptDate>
                    <ns11:quantity xsi:type="xsd:double">3.0</ns11:quantity>
                    <ns11:orderType xsi:type="ns15:ItemSupplyPlanOrderType" xmlns:ns15="urn:ty
pes.demandplanning_2017_1.transactions.webservices.netsuite.com">_purchaseOrder</ns11:orderType
>

                 </ns11:itemSupplyPlanOrder>
              </ns11:orderList>
           </record>
        </addList>
     </soapenv:Body>

   */
```

Java

```
public void addListItemSupplyPlan() throws Exception
   {
      this.login(true);

     ItemSupplyPlan[] isp = new ItemSupplyPlan[2];

     ItemSupplyPlanOrderList ispl = new ItemSupplyPlanOrderList();
      ItemSupplyPlanOrder[] ispo =  new ItemSupplyPlanOrder[2];

      ispo[0] = new ItemSupplyPlanOrder();

       Calendar c = Calendar.getInstance();
       c.set(2012, 5, 12);
      ispo[0].setOrderDate(c);

      Calendar c2 = Calendar.getInstance();
      c2.set(2012, 5, 12);
      ispo[0].setReceiptDate(c2);

        ispo[0].setQuantity(3.0);
       ispo[0].setOrderType(ItemSupplyPlanOrderType._purchaseOrder);

        ispo[1] = new ItemSupplyPlanOrder();

        Calendar c3 = Calendar.getInstance();
      c3.set(2012, 6, 05);
      ispo[1].setOrderDate(c3);

      Calendar c4 = Calendar.getInstance();
```

```
        c4.set(2012, 6, 13);
        ispo[1].setReceiptDate(c4);

         ispo[1].setQuantity(3.0);
         ispo[1].setOrderType(ItemSupplyPlanOrderType._purchaseOrder);


         RecordRef[] rrItem = new RecordRef[2];
         rrItem[0] =  new RecordRef();
        rrItem[0].setInternalId("21");
        rrItem[0].setType(RecordType.inventoryItem);

        rrItem[1] =  new RecordRef();
        rrItem[1].setInternalId("22");
        rrItem[1].setType(RecordType.inventoryItem);


       for(int i = 0; i < 2; i++){

           isp[i] = new ItemSupplyPlan();
           RecordRef rrSubsidiary = new RecordRef();
           rrSubsidiary.setInternalId("1");
           rrSubsidiary.setType(RecordType.subsidiary);
           isp[0].setSubsidiary(Util.makeRecordRef("1"));

           RecordRef rrLocation = new RecordRef();
           rrLocation.setInternalId("1");
           rrLocation.setType(RecordType.location);
           isp[i].setLocation(rrLocation);


           isp[i].setItem(rrItem[i]);

             ispl.setItemSupplyPlanOrder(ispo);
             isp[i].setOrderList(ispl);
       }

        _port.add(isp);
     }
```

Each of the following examples uses the update operation.

SOAP Request

```
/*

   <update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record externalId="Strawberry10442165" internalId="6" xsi:type="ns8:ItemSupplyPlan
" xmlns:ns8="urn:demandplanning_2017_1.transactions.webservices.netsuite.com">
               <ns8:location internalId="1" xsi:type="ns9:RecordRef" xmlns:ns9="urn:core_2017_1
.platform.webservices.netsuite.com">
                  <ns9:name xsi:type="xsd:string">LOC 1</ns9:name>
               </ns8:location>
               <ns8:item internalId="22" xsi:type="ns10:RecordRef" xmlns:ns10="urn:core_2017_1.
platform.webservices.netsuite.com">
                  <ns10:name xsi:type="xsd:string">Item2-ISP</ns10:name>
               </ns8:item>
               <ns8:memo xsi:type="xsd:string">IamupdatingthisISP</ns8:memo>
               <ns8:orderList replaceAll="false" xsi:type="ns8:ItemSupplyPlanOrderList">
                  <ns8:itemSupplyPlanOrder xsi:type="ns8:ItemSupplyPlanOrder">
                     <ns8:orderLineId xsi:type="xsd:long">0</ns8:orderLineId>
                     <ns8:orderDate xsi:type="xsd:dateTime">2012-06-11T07:00:00.000Z</ns8:order
Date>
                     <ns8:receiptDate xsi:type="xsd:dateTime">2012-06-11T07:00:00.000Z</ns8:rec
eiptDate>
                     <ns8:quantity xsi:type="xsd:double">3.0</ns8:quantity>
                     <ns8:orderCreated xsi:type="xsd:boolean">false</ns8:orderCreated>
                     <ns8:orderType xsi:type="ns11:ItemSupplyPlanOrderType" xmlns:ns11="urn:typ
es.demandplanning_2017_1.transactions.webservices.netsuite.com">_purchaseOrder</ns8:orderType>

                  </ns8:itemSupplyPlanOrder>
                  <ns8:itemSupplyPlanOrder xsi:type="ns8:ItemSupplyPlanOrder">
                     <ns8:orderLineId xsi:type="xsd:long">1</ns8:orderLineId>
                     <ns8:orderDate xsi:type="xsd:dateTime">2012-07-04T07:00:00.000Z</ns8:order
Date>
                     <ns8:receiptDate xsi:type="xsd:dateTime">2012-07-13T16:54:01.528Z</ns8:rec
eiptDate>
                     <ns8:quantity xsi:type="xsd:double">3.0</ns8:quantity>
                     <ns8:orderCreated xsi:type="xsd:boolean">false</ns8:orderCreated>
                     <ns8:orderType xsi:type="ns12:ItemSupplyPlanOrderType" xmlns:ns12="urn:typ
es.demandplanning_2017_1.transactions.webservices.netsuite.com">_purchaseOrder</ns8:orderType>

                  </ns8:itemSupplyPlanOrder>
               </ns8:orderList>
            </record>
         </update>

   */
```

Java

```
public void  updateItemSupplyPlan() throws Exception
   {
      this.login(true);

      RecordRef rrIsp =  new RecordRef();
      rrIsp.setInternalId("6");
      rrIsp.setType(RecordType.itemSupplyPlan);

      ItemSupplyPlan isp =  (ItemSupplyPlan)sessMgr.getPort().get(rrIsp).getRecord();

      isp.setMemo("IamupdatingthisISP");
      Calendar c = Calendar.getInstance();
      c.set(2012, 6, 14);
```

```
        isp.getOrderList().getItemSupplyPlanOrder(1).setReceiptDate(c);

        Preferences p =  new Preferences();
        p.setIgnoreReadOnlyFields(true);

        _setPreferences(p);
        _Port().update(isp);
    }
```

Each of the following examples uses the search operation.

SOAP Request

```
/*
   Request
    <soapenv:Body>
         <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <searchRecord xsi:type="ns6:ItemSupplyPlanSearch" xmlns:ns6="urn:demandplanning_201
7_1.transactions.webservices.netsuite.com">
               <ns6:basic xsi:type="ns7:ItemSupplyPlanSearchBasic" xmlns:ns7="urn:common_2017_1
.platform.webservices.netsuite.com">
                  <ns7:item operator="anyOf" xsi:type="ns8:SearchMultiSelectField" xmlns:ns8="u
rn:core_2017_1.platform.webservices.netsuite.com">
                     <ns8:searchValue internalId="22" type="inventoryItem" xsi:type="ns8:Record
Ref"/>
                  </ns7:item>
               </ns6:basic>
            </searchRecord>
         </search>
      </soapenv:Body>

   */
```

Java

```
public void searchItemSupplyPlan() throws Exception
    {
         this.login(true);
        ItemSupplyPlanSearch isp =  new ItemSupplyPlanSearch();
      ItemSupplyPlanSearchBasic ispb =  new  ItemSupplyPlanSearchBasic();

      RecordRef rrItem =  new RecordRef();
      rrItem =  new RecordRef();
      rrItem.setInternalId("22");
      rrItem.setType(RecordType.inventoryItem);

      SearchMultiSelectField smsi =  new SearchMultiSelectField();
      smsi.setOperator(SearchMultiSelectFieldOperator.anyOf);
      smsi.setSearchValue(new RecordRef[] { rrItem });
      ispb.setItem(smsi);
       isp.setBasic(ispb);
       SearchResult sr = _port.search(isp);
   }
```

Journal Entry

You use the journal entry record to adjust balances in accounts. Journal entries let you change the value of any set of accounts without having to enter a posting transaction. In an account that has the Multi-Book Accounting feature enabled, you can also use this record to create book specific journal entries.

In the user interface, you access this record as follows:

- **Journal entries** – Go to Transactions > Financial > Make Journal Entries. For help working with this record in the user interface, see the help topic Making Journal Entries.
- **Book specific journal entries** – Go to Transactions > Financial > Make Book Specific Journal Entries. Note that this form is available only for accounts that use Multi-Book Accounting. For help working with this record in the user interface, see the help topic Book-Specific Journal Entries.

The journal entry record is defined in the tranGeneral XSD, where it is called JournalEntry.

For more information about using SuiteTalk to work with the journal entry record, see the following sections:

- Journal Entry Supported Operations
- Journal Entry Body and Sublist Fields
- Journal Entry Code Samples
- Common Errors With Journal Entries

Journal Entry Supported Operations

The following operations are supported for use with the journal entry record:

add | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

For examples of how to use the add operation, see Adding a Journal Entry and Adding a Book Specific Journal Entry.

Journal Entry Body and Sublist Fields

Be aware that the way you interact with certain fields varies depending on whether the journal entry is book specific. For more information, see the following sections:

- Body Fields

- Sublist Fields

Note: For a full list of available fields, refer to the journal entry reference page in the SuiteTalk Schema Browser. For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Body Fields

The following table describes several key fields on the journal entry record. The way you use these fields varies depending on whether the journal entry is book specific. The guidance in this table assumes that you are adding a new journal entry record.

Field	Label in UI	If you want to create a regular journal entry (not book specific)	If you want to create a book specific journal entry
accountingBook	Accounting Book	Do not use this field. Using this field automatically makes the journal entry book specific.	Name the accounting book that this journal entry is for.
currency	Currency	Valid values include any currency defined for your account at Lists > Accounting > Currencies.	Do not set a value for this field. The system automatically chooses the correct value. This choice is based on the combination of two values on the body of the book specific journal entry record: - accountingBook - subsidiary The currency value is defined on the Currencies sublist of the accounting book record.
exchangeRate	Exchange Rate	This field reflects the exchange rate between: - The value in the journal entry's currency field. - The base currency for the subsidiary named in the journal entry's subsidiary field. You can leave this field empty to let the system fill in the value, or you can specify an exchange rate.	This value is always 1.0. If you try to set a value for this field, your input is ignored. This is because the currency for the entry must be the default used for this combination of accounting book and subsidiary. It cannot be a different currency that requires conversion.
isBookSpecific	This field is hidden in the UI.	Omit a value for this field. The system will automatically assign this field the correct value.	If you are actively setting a value for the accountingBook field, you can omit a value for isBookSpecific. The system will

Field	Label in UI	If you want to create a regular journal entry (not book specific)	If you want to create a book specific journal entry
		(The correct value in this case is false.)	automatically assign this field the correct value. (The correct value in this case is true.) As an alternative, you can set isBookSpecific to true and omit a value for accounting book. This approach creates a book specific journal entry for the primary accounting book.
memo	Memo	This field is available only with the 2015.2 WSDL and later. By contrast, the Memo field from the Lines sublist is available in all supported WSDLs.	
subsidiary	Subsidiary	Choose one of your account's subsidiaries (listed at Setup > Company > Subsidiaries).	Valid values for subsidiary are determined by the value of the accountingBook field. You can choose only subsidiaries that are listed on the Currencies subtab of the selected accounting book record.

Sublist Fields

The line sublist is the main sublist on the record. It is used to record credits and debits. Every journal entry must have at least two lines in the line sublist: one credit and one debit. The total value of credits must equal the total value of debits.

Note: As of the 2016.1 WSDL, another sublist, the accountingBookDetailList, is also exposed to web services. This sublist is relevant only if you use the Multi-Book Accounting feature. For details on this sublist, see the SuiteTalk Schema Browser's accountingBookDetailList reference page.

Revenue Recognition Fields

Several fields on the line sublist are used only when the Revenue Recognition feature is enabled. Further, these fields are used only for journal entries that are not book specific.

These fields include the following:

- endDate
- residual
- schedule
- startDate

Maximum Number of Lines

You must have at least two line items to submit a journal entry transaction. The maximum number of lines per journal entry is **1,000** when using synchronous web services processing. With asynchronous web services processing, the limit is 10,000 lines. If your import includes journal entries with more than 1,000 lines, CSV import can also serve as an alternative. For journal entry transactions submitted through CSV import, the limit is 10,000 lines per transaction.

Journal Entry Code Samples

For examples of how to structure your journal entry code, refer to the following sections:

- Adding a Journal Entry
- Adding a Book Specific Journal Entry

Adding a Journal Entry

The following example shows how to create a journal entry record that has two sublist lines.

C#

```
private void addJournalEntry()
      {

          JournalEntry newJournalEntry = new JournalEntry();
          newJournalEntry.externalId = "201A";

          RecordRef mySubsidiary = new RecordRef();
          mySubsidiary.internalId = "1";
          newJournalEntry.subsidiary = mySubsidiary;

          RecordRef myCurrency = new RecordRef();
          myCurrency.internalId = "3";
          newJournalEntry.currency = myCurrency;

          newJournalEntry.exchangeRate = .911;
          newJournalEntry.exchangeRateSpecified = true;

          RecordRef myDebitAccount = new RecordRef();
          myDebitAccount.internalId = "7";

          RecordRef myCreditAccount = new RecordRef();
          myCreditAccount.internalId = "8";

          newJournalEntry.lineList = new JournalEntryLineList();
          newJournalEntry.lineList.line = new JournalEntryLine[2];

          newJournalEntry.lineList.line[0] = new JournalEntryLine();
          newJournalEntry.lineList.line[0].account = myDebitAccount;
          newJournalEntry.lineList.line[0].debit = 1000;
          newJournalEntry.lineList.line[0].debitSpecified = true;

          newJournalEntry.lineList.line[1] = new JournalEntryLine();
          newJournalEntry.lineList.line[1].account = myCreditAccount;
          newJournalEntry.lineList.line[1].credit = 1000;
          newJournalEntry.lineList.line[1].creditSpecified = true;

          _service.add(newJournalEntry);
```

```
        }
```

SOAP Request

```
<soap:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record xmlns:q1="urn:general_2017_1.transactions.webservices.netsuite.com" xsi:type="q1:
JournalEntry" externalId="201A">
         <q1:currency internalId="3" />
         <q1:exchangeRate>0.911</q1:exchangeRate>
         <q1:subsidiary internalId="1" />
         <q1:lineList>
            <q1:line>
               <q1:account internalId="7" />
               <q1:debit>1000</q1:debit>
            </q1:line>
            <q1:line>
               <q1:account internalId="8" />
               <q1:credit>1000</q1:credit>
            </q1:line>
         </q1:lineList>
      </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 internalId="8842" externalId="201A" type="journalEntry" xsi:type="platformCor
e:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Adding a Book Specific Journal Entry

The following example shows how to create a book specific journal entry record that has two sublist lines.

Note that with this type of journal entry, you should not set a value for the currency field. There is only one valid currency for a book specific journal entry. The system automatically populates this field. The value is based on the values you choose for the journal entry's subsidiary and accounting book fields.

C#

```
private void addBookSpecificJournalEntry()
```

```
   {

      JournalEntry newJournalEntry = new JournalEntry();
      newJournalEntry.externalId = "105A";

      RecordRef mySubsidiary = new RecordRef();
      mySubsidiary.internalId = "10";
      newJournalEntry.subsidiary = mySubsidiary;

      RecordRef myAccountingBook = new RecordRef();
      myAccountingBook.internalId = "2";
      newJournalEntry.accountingBook = myAccountingBook;

      RecordRef myDebitAccount = new RecordRef();
      myDebitAccount.internalId = "214";

      RecordRef myCreditAccount = new RecordRef();
      myCreditAccount.internalId = "215";

      newJournalEntry.lineList = new JournalEntryLineList();
      newJournalEntry.lineList.line = new JournalEntryLine[2];

      newJournalEntry.lineList.line[0] = new JournalEntryLine();
      newJournalEntry.lineList.line[0].account = myDebitAccount;
      newJournalEntry.lineList.line[0].debit = 21000;
      newJournalEntry.lineList.line[0].debitSpecified = true;

      newJournalEntry.lineList.line[1] = new JournalEntryLine();
      newJournalEntry.lineList.line[1].account = myCreditAccount;
      newJournalEntry.lineList.line[1].credit = 21000;
      newJournalEntry.lineList.line[1].creditSpecified = true;

      _service.add(newJournalEntry);

}
```

SOAP Request

```
<soap:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record externalId="105A" xsi:type="q1:JournalEntry" xmlns:q1="urn:general_2017_1.transac
tions.webservices.netsuite.com">
         <q1:accountingBook internalId="2"/>
         <q1:subsidiary internalId="10"/>
         <q1:lineList>
            <q1:line>
               <q1:account internalId="214"/>
               <q1:debit>21000</q1:debit>
            </q1:line>
            <q1:line>
               <q1:account internalId="215"/>
               <q1:credit>21000</q1:credit>
            </q1:line>
         </q1:lineList>
      </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="journalEntry" externalId="105A" inter
nalId="18" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Common Errors With Journal Entries

When you are working with journal entry records, you may see following failure messages in your SOAP responses.

This record already exists

This error indicates indicates that you attempted to add a journal entry using an external ID that is already in use.

Invalid currency reference key x for accountingbook y

This error indicates that you attempted to set an incorrect value for the currency field of a book specific journal entry. In general, you should not try to set a value for currency. The system automatically populates this field based on the value of other fields on the record.

Invalid account reference key x for subsidiary y

Valid values for the account field may vary. Valid options are determined by the value of the subsidiary body field. If you see this error, review the data and make sure that you have chosen appropriate account values. For information about body and sublist fields, see Journal Entry Body and Sublist Fields.

The amounts in a journal entry must balance

The lines of a journal entry must include at least one debit and at least one credit. The total debit value must equal the total credit value. This error indicates that they do not match.

Please enter a value for subsidiary

A value for subsidiary is always required, even when there is only one valid choice.

Note: For information about other possible errors with journal entries, search for the error message in SuiteAnswers.

Manufacturing Operation Task

If the Manufacturing Routing and Work Center feature has been enabled, you can use SuiteTalk to work with manufacturing operation task records. You can check to see whether the feature is enabled by going to Setup > Company > Enable Features, and reviewing the Items & Inventory tab.

The Manufacturing Routing and Work Center feature lets you specify a sequence of tasks required for the completion of a Work In Process (WIP) work order. This record represents a job that must be completed by a specific employee group. In the UI, generally these records are created automatically when you save a WIP work order that references a specific routing record — each step described in the routing record becomes an operation task record, viewable on the work order's Operations subtab. You can also manually create an operation task record by clicking the New Operation Task button on the work order's Operations subtab. You can view all existing manufacturing operation task records by going to Transactions > Manufacturing > Manufacturing Operation Tasks.

For more details about this record, see the help topic Manufacturing Operation Tasks. For more information about the Manufacturing Routing and Work Center feature, see the help topic Manufacturing Routing.

The manufacturing operation task record is defined in the listScm XSD.

Supported Operations

The following operations can be used with the operation task record.

add | addList | delete | deleteList | get |getDeleted |getList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s manufacturing operation task reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Refer to the following sections for more details on working with manufacturing operation task records.

Prerequisites for Creating a Record

Before you can add a manufacturing operation task record, a WIP work order that references a routing record must already exist in your system. To create an operation task record, you must reference this work order using the workOrder element.

Required Fields

Be aware of the following characteristics of this record's required fields:

- Operation Sequence — The numeric value entered for this field must be greater than the largest Operation Sequence value already listed on the work order's Operations subtab. You set this field using the operationSequence element.
- Operation Name — In web services, you set this value using the title element.

Other fields are also required. It may be useful to refer to Editing a Manufacturing Operation Task for more details about the record and its required fields.

Sample Code

The following code illustrates how to add a manufacturing operation task record.

Java

```
public void testAddOperationTask() throws Exception{
// This operation requires a valid session
this.login(true);

ManufacturingOperationTask manufacturingOperationTask = new ManufacturingOperationTask();
RecordRef workOrderRef = new RecordRef();
workOrderRef.setInternalId("65");
manufacturingOperationTask.setWorkOrder(workOrderRef);
manufacturingOperationTask.setTitle("Task title");
manufacturingOperationTask.setOperationSequence(3L);
manufacturingOperationTask.setSetupTime(30.0);
manufacturingOperationTask.setRunRate(20.0);
RecordRef costTemplateRef = new RecordRef();
costTemplateRef.setInternalId("1");
manufacturingOperationTask.setManufacturingCostTemplate(costTemplateRef);
RecordRef workCenterRef = new RecordRef();
workCenterRef.setInternalId("113");
manufacturingOperationTask.setManufacturingWorkCenter(workCenterRef);
manufacturingOperationTask.setMachineResources(14L);
manufacturingOperationTask.setLaborResources(17L);
c.addRecord(manufacturingOperationTask);
}
```

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: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">nlbuild@ne
tsuite.com</ns2:email>
            <ns3:password xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">passwor
d</ns3:password>
            <ns4:account xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com">3604360<
/ns4:account>
            <ns5:role internalId="3" xmlns:ns5="urn:core_2017_1.platform.webservices.netsuite.c
om"/>
         </ns1:passport>
         <ns6:preferences soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next" soapenv:mu
stUnderstand="0" xmlns:ns6="urn:messages_2017_1.platform.webservices.netsuite.com">
            <ns6:ignoreReadOnlyFields>true</ns6:ignoreReadOnlyFields>
         </ns6:preferences>
      </soapenv:Header>
      <soapenv:Body>
         <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record xsi:type="ns7:ManufacturingOperationTask" xmlns:ns7="urn:supplychain_2017_1
.lists.webservices.netsuite.com">
               <ns7:manufacturingWorkCenter internalId="113" xsi:type="ns8:RecordRef" xmlns:ns8
="urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns7:manufacturingCostTemplate internalId="1" xsi:type="ns9:RecordRef" xmlns:ns9
="urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns7:title xsi:type="xsd:string">Task title</ns7:title>
               <ns7:operationSequence xsi:type="xsd:long">3</ns7:operationSequence>
               <ns7:workOrder internalId="65" xsi:type="ns10:RecordRef" xmlns:ns10="urn:core_20
17_1.platform.webservices.netsuite.com"/>
               <ns7:setupTime xsi:type="xsd:double">30.0</ns7:setupTime>
               <ns7:runRate xsi:type="xsd:double">20.0</ns7:runRate>
               <ns7:machineResources xsi:type="xsd:long">14</ns7:machineResources>
               <ns7:laborResources xsi:type="xsd:long">17</ns7:laborResources>
            </record>
         </add>
      </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:Header>
         <platformMsgs:documentInfo xmlns:platformMsgs="urn:messages_2017_1.platform.webservice
s.netsuite.com">
            <platformMsgs:nsId>WEBSERVICES_3604360_0311201314979673951785156835_4047dcfafa5</pl
atformMsgs:nsId>
         </platformMsgs:documentInfo>
      </soapenv:Header>
      <soapenv:Body>
```

```
         <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <writeResponse>
               <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
               <baseRef internalId="716" type="manufacturingOperationTask" xsi:type="platformCo
re:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </writeResponse>
         </addResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Opportunity

Opportunities represent negotiations with prospects. You must first enable opportunities in your NetSuite account before you can access this record type. To enable opportunities, go to Setup > Company > Enable Features. On the CRM tab, under Sales, select the Opportunities check box.

The opportunity record is defined in the tranSales (sales) XSD.

Supported Operations

The following operations can be used with the opportunity record.

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's opportunity reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

For details about working with opportunities in NetSuite, see the help topic Opportunity Records.

Associating Transactions with Opportunities

You can associate estimates, cash sales, sales orders, and invoices with opportunities. After a transaction other than an estimate is associated with an opportunity, the opportunity's status is automatically set to **Closed Won**. After an opportunity's status is set to **Closed Won**, it is no longer available to be selected on other cash sale, sales order, or invoice records.

Projected Total, Range High and Range Low Fields

When working with opportunity records in the UI, the range low, range high and projected total values must comply with the following rules:

- The rangeLow value must be lower than or equal to the projectedTotal value. This value represents the Worst Case projected total.
- The rangeHigh value must be greater than or equal to the projectedTotal value. This amount represents the Upside projected total.

In the UI, these values are calculated but can be overridden by the user. The calculated values are determined by relationships between the summation of amounts of entries in the itemList, projectedTotal values, and range low / range high values. When any of the values are overridden by the user, client side validation ensures that the fields are set correctly according to the above rules and automatically recalculates where appropriate.

To maintain proper rangeHigh, rangeLow and projectedTotal relationships when using web services, the behavior is as follows:

If Advanced Forecasting is Off

On Add

- projectedTotal is required unless there is an item in the item list.
- If projectedTotal is provided and there are items, then projectedTotal is calculated (with the sum of the item amounts).
- If projectedTotal is provided then it is set unless it is equal to the sum of all items, where in that case it is calculated.

On update

- If projectedTotal is not provided and there are no items in the update, then projectedTotal is not changed.
- If projectedTotal is not provided and there are items in the update request, then projectedTotal is set to the sum of all items.
- If projectedTotal is not provided and there are items in the original but no items in the update request, then projectedTotal is not changed except where projectedTotal is equal to the sum of all item amounts. In that case it is calculated.
- If projectedTotal is provided and there are items in the original, then projectedTotal is set.
- If projectedTotal is provided are there are items in the update request, then projectedTotal is set except where projectedTotal is equal to the sum of all items in the update request. In the case it is calculated.

If Advanced Forecasting is On

On Add

- If no item list is sent in the request and projectedTotal is not provided an error will be thrown.
- If no item list is sent and one or both of rangeLow or rangeHigh are missing then both are set to projectedTotal
- If a value is NOT provided for projectedTotal, rangeLow or rangeHigh fields, then each field is calculated.
- If a value is submitted for any one field then that field is NOT calculated except when a projectedTotal value is equal to the sum of the amounts for entries in the itemList at any point during item summation. In this case the projectedTotal value is calculated (set to the sum of all item amounts) even though a value was submitted.

On Update

- If any of the three fields has the same value as previously set, whether calculated or not, all three fields are calculated.
- If all three fields have different values they are not calculated except when a projectedTotal value is equal to the sum of the amounts for entries in the itemList at any point during item summation. In this case the projectedTotal value is calculated (set to the sum of all item amounts) even though a value was submitted.
- If any of the three fields is empty, then all three fields are populated with calculated values except when there is no change to the itemList.

Working with Opportunity Sublists

The SuiteTalk Schema Browser includes all sublists associated with the opportunity record. See the following information for usage notes regarding specific Opportunity sublists. Usage notes are not provided for every sublist type.

- OpportunityItemList
- OpportunitySalesTeam

OpportunityItemList

In an opportunity, a list of items can be added, modified or deleted in the items sublist. However, each item listed in the item sublist is not a keyed entry — the list is an array of items. Therefore, to modify the contents of the items sublist for a particular opportunity record, follow these guidelines:

- To update the list of items, resubmit the **entire** list (array) of items associated with the opportunity record.
- To delete a subset of items on the list, submit a partial list of what was retrieved with the get operation.
- To delete the entire list, submit an empty list.

Important: If you are using a customForm that has location, department and class customized at the item level, you can NOT set these fields at the body level.

OpportunitySalesTeam

This list is only available when the Team Selling feature is enabled.

Paycheck

The paycheck record is available when the Payroll feature is enabled on the Employees subtab at Setup > Company > Setup Tasks > Enable Features.

In the user interface, you can access the paycheck record at Transactions > Employees > Create Payroll.

For information about working with this record in the user interface, see the help topic Editing an Individual Paycheck from a Payroll Batch.

The paycheck record is defined in the tranEmp (employees) XSD.

Supported Operations

The following operations are supported with the paycheck record.

delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue |search | searchMore | searchMoreWithId | searchNext |update | updateList | upsert (update only)| upsertList (update only)

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's paycheck reference page.

Usage Notes

- The paycheck record is created through the payroll batch record when you add employees to the payroll batch. You cannot create paycheck records through web services, you can only update paychecks that already exist in a payroll batch.
- The employee whose paycheck data you want to update must be included in the payroll batch, and the payroll item you want to add to a paycheck must be available for that employee.
- You cannot update paycheck data in a payroll batch that had already been committed.

Paycheck Journal

The paycheck journal transaction serves as a payroll interface for web services integrations with external payroll systems. This transaction is similar to the paycheck transaction used by the NetSuite Payroll feature. However, unlike a paycheck, a paycheck journal cannot issue a check drawn on an account. On its own, a paycheck journal functions as a journal entry; it records accounting impact.

This transaction is available when the Paycheck Journal feature is enabled at Setup > Company > Enable Features, on the Employees subtab.

For more details about this type of transaction, see the help topic Using the Paycheck Journal Feature.

The paycheck journal record is defined in the tranEmp (employees) XSD.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Supported Operations

The following operations can be used with paycheck journal records:

add | addList | delete | deleteList | get | getDeleted | getList | search | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's paycheck journal reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Working with Paycheck Journal Sublists

This record includes the following sublists:

- PaycheckJournalCompanyContributionList
- PaycheckJournalDeductionList
- PaycheckJournalEmployeeTaxList
- PaycheckJournalCompanyTaxList
- PaycheckJournalEarningList
- CustomFieldList

The CustomFieldList sublist stores values for custom transaction body fields added to a paycheck journal. Note that each of the other sublists can include its own CustomFieldList to store values for custom transaction column fields.

Sample Code

The following examples add a paycheck journal transaction.

SOAP Request

```
/*
   <soapenv:Body>
         <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record externalId="Grape14276260" xsi:type="ns6:PaycheckJournal" xmlns:ns6="urn:em
ployees_2017_1.transactions.webservices.netsuite.com">
               <ns6:employee internalId="4" type="employee" xsi:type="ns7:RecordRef" xmlns:ns7=
"urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns6:tranDate xsi:type="xsd:dateTime">2012-06-10T14:33:56.451Z</ns6:tranDate>

               <ns6:account internalId="4" type="account" xsi:type="ns8:RecordRef" xmlns:ns8="u
rn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns6:earningList replaceAll="false" xsi:type="ns6:PaycheckJournalEarningList">

                  <ns6:paycheckJournalEarning xsi:type="ns6:PaycheckJournalEarning">
                     <ns6:payrollItem internalId="102" type="payrollItem" xsi:type="ns9:RecordR
ef" xmlns:ns9="urn:core_2017_1.platform.webservices.netsuite.com"/>
                     <ns6:amount xsi:type="xsd:double">20.0</ns6:amount>
                  </ns6:paycheckJournalEarning>
                  <ns6:paycheckJournalEarning xsi:type="ns6:PaycheckJournalEarning">
                     <ns6:payrollItem internalId="102" type="payrollItem" xsi:type="ns10:Record
Ref" xmlns:ns10="urn:core_2017_1.platform.webservices.netsuite.com"/>
                     <ns6:amount xsi:type="xsd:double">20.0</ns6:amount>
                  </ns6:paycheckJournalEarning>
               </ns6:earningList>
            </record>
         </add>
      </soapenv:Body>
     */
```

Java

```
public void addPaycheckJournalSample() throws Exception
    {
       // This operation requires a valid session
       this.login(true);

       PaycheckJournal pj =  new PaycheckJournal();

       RecordRef rrEmployee =  new RecordRef();
       rrEmployee.setType(RecordType.employee);
       rrEmployee.setInternalId("4");

       pj.setEmployee(rrEmployee);

       RecordRef rrAccount =  new RecordRef();
       rrAccount.setType(RecordType.account);
       rrAccount.setInternalId("5");

        pj.setAccount(rrAccount);

        Calendar c  = Calendar.getInstance();
        c.set(2012, 5, 10); //June 10, 2012
        pj.setTranDate(c);

        //setting 2 lines of Payroll Item Earning Type
        PaycheckJournalEarningList pjel =  new PaycheckJournalEarningList();
        PaycheckJournalEarning[] pje =  new PaycheckJournalEarning[2];

        RecordRef rrPIEarning =  new RecordRef();
        rrPIEarning.setType(RecordType.payrollItem);
        rrPIEarning.setInternalId("102");

        RecordRef rrPIDeduction =  new RecordRef();
        rrPIDeduction.setType(RecordType.payrollItem);
        rrPIDeduction.setInternalId("103");

        pje[0] =  new PaycheckJournalEarning();
        pje[0].setPayrollItem(rrPIEarning);
        pje[0].setAmount(20.00);

        pje[1] =  new PaycheckJournalEarning();
        pje[1].setPayrollItem(rrPIEarning);
        pje[1].setAmount(20.00);

        pjel.setPaycheckJournalEarning(pje);
        pj.setEarningList(pjel);

        //setting 1 line of Payroll Item Deduction Type
        PaycheckJournalDeductionList pjdl =  new PaycheckJournalDeductionList();
        PaycheckJournalDeduction[] pjd = new PaycheckJournalDeduction[1];
        pjd[0] = new PaycheckJournalDeduction();
        pjd[0].setAmount(30.00);
        pjd[0].setPayrollItem(rrPIDeduction);
        pjdl.setPaycheckJournalDeduction(pjd);

        _port.add(pi);
    }
```

The following examples update a paycheck journal transaction.

SOAP Request

```
/* get SOAP Request
   <soapenv:Body>
         <get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <baseRef internalId="102" type="paycheckJournal" xsi:type="ns6:RecordRef" xmlns:ns6
="urn:core_2017_1.platform.webservices.netsuite.com"/>
         </get>
    </soapenv:Body>
   */

   /* upate SOAP Request
    <record externalId="Pear11044478" xsi:type="ns6:PaycheckJournal" xmlns:ns6="urn:employees_2
017_1.transactions.webservices.netsuite.com">
               <ns6:currency internalId="1" xsi:type="ns7:RecordRef" xmlns:ns7="urn:core_2017_1
```

```
.platform.webservices.netsuite.com">
                  <ns7:name xsi:type="xsd:string">Canadian dollar</ns7:name>
               </ns6:currency>
               <ns6:exchangeRate xsi:type="xsd:double">1.0</ns6:exchangeRate>
               <ns6:createdDate xsi:type="xsd:dateTime">2012-07-26T07:33:41.000Z</ns6:createdDa
te>
               <ns6:lastModifiedDate xsi:type="xsd:dateTime">2012-07-26T07:33:41.000Z</ns6:last
ModifiedDate>
               <ns6:employee internalId="4" xsi:type="ns8:RecordRef" xmlns:ns8="urn:core_2017_1
.platform.webservices.netsuite.com">
                  <ns8:name xsi:type="xsd:string">Mariah  Carey</ns8:name>
               </ns6:employee>
               <ns6:tranDate xsi:type="xsd:dateTime">2012-06-10T07:00:00.000Z</ns6:tranDate>

               <ns6:postingPeriod internalId="77" xsi:type="ns9:RecordRef" xmlns:ns9="urn:core_
2017_1.platform.webservices.netsuite.com">
                  <ns9:name xsi:type="xsd:string">Jun 2012</ns9:name>
               </ns6:postingPeriod>
               <ns6:account internalId="4" xsi:type="ns10:RecordRef" xmlns:ns10="urn:core_2017_
1.platform.webservices.netsuite.com">
                  <ns10:name xsi:type="xsd:string">Petty Cash</ns10:name>
               </ns6:account>
               <ns6:earningList replaceAll="false" xsi:type="ns6:PaycheckJournalEarningList">

                  <ns6:paycheckJournalEarning xsi:type="ns6:PaycheckJournalEarning">
                     <ns6:id xsi:type="xsd:long">1</ns6:id>
                     <ns6:payrollItem internalId="102" xsi:type="ns11:RecordRef" xmlns:ns11="ur
n:core_2017_1.platform.webservices.netsuite.com">
                        <ns11:name xsi:type="xsd:string">Re-Employment Service Fund</ns11:name>

                     </ns6:payrollItem>
                     <ns6:amount xsi:type="xsd:double">99.99</ns6:amount>
                  </ns6:paycheckJournalEarning>
                  <ns6:paycheckJournalEarning xsi:type="ns6:PaycheckJournalEarning">
                     <ns6:id xsi:type="xsd:long">2</ns6:id>
                     <ns6:payrollItem internalId="102" xsi:type="ns12:RecordRef" xmlns:ns12="ur
n:core_2017_1.platform.webservices.netsuite.com">
                        <ns12:name xsi:type="xsd:string">Re-Employment Service Fund</ns12:name>

                     </ns6:payrollItem>
                     <ns6:amount xsi:type="xsd:double">20.0</ns6:amount>
                  </ns6:paycheckJournalEarning>
               </ns6:earningList>
            </record>
   */
```

Java

```
  public void updatePaycheckJournalSample() throws Exception
    {
       this.login(true);

       RecordRef rrPaycheck =  new RecordRef();
       rrPaycheck.setInternalId("102");
       rrPaycheck.setType(RecordType.paycheckJournal);

       ReadResponse rr = (PaycheckJournal)_port.get(rrPaycheck);
      PaycheckJournal pj = rr.getRecord();

       pj.getEarningList().getPaycheckJournalEarning(0).setAmount(99.99);
       _port.update(pj);
    }
```

The following examples do a basic search for paycheck journal transactions with a date of June 10, 2012.

SOAP Request

```
/*

     <soapenv:Body>
         <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <searchRecord xsi:type="ns6:TransactionSearchBasic" xmlns:ns6="urn:common_2017_1.pl
atform.webservices.netsuite.com">
               <ns6:tranDate operator="on" xsi:type="ns7:SearchDateField" xmlns:ns7="urn:core_2
017_1.platform.webservices.netsuite.com">
                  <ns7:searchValue xsi:type="xsd:dateTime">2012-06-10T14:55:12.290Z</ns7:search
Value>
               </ns6:tranDate>
               <ns6:type operator="anyOf" xsi:type="ns8:SearchEnumMultiSelectField" xmlns:ns8="
urn:core_2017_1.platform.webservices.netsuite.com">
                  <ns8:searchValue xsi:type="xsd:string">_paycheckJournal</ns8:searchValue>

               </ns6:type>
            </searchRecord>
         </search>
      </soapenv:Body>

   */
```

Java

```
public void searchPaycheckJournal() throws Exception
    {
       this.login(true);

       TransactionSearchBasic tsb =  new TransactionSearchBasic();
       SearchEnumMultiSelectField semsf = new SearchEnumMultiSelectField();
       semsf.setOperator(SearchEnumMultiSelectFieldOperator.anyOf);
       semsf.setSearchValue(new String[] {"_paycheckJournal"});
       tsb.setType(semsf);

       SearchDateField sdf = new SearchDateField();
```

```
        sdf.setOperator(SearchDateFieldOperator.on);

        Calendar c =  Calendar.getInstance();
        c.set(2012, 5, 10); //June 10, 2012
        sdf.setSearchValue(c);

        tsb.setTranDate(sdf);
        sessMgr.getPort().search(tsb);
    }
```

Purchase Order

A purchase order transaction authorizes the purchase of goods and/or services. This transaction tracks items expected to be received, items actually received, and items yet to be received. A purchase order has no accounting impact until the included items are received. This transaction is enabled when the Purchase Orders feature is enabled.

For more details about this type of transaction, see the help topic Purchasing.

The purchase order record is defined in the tranPurch (purchases) XSD.

Supported Operations

The following operations can be used with purchase order records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's purchase order reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Working with Purchase Order Sublists

The SuiteTalk Schema Browser includes all sublists associated with the purchase order record. See the following information for usage notes regarding specific purchase order sublists. Usage notes are not provided for every list type.

PurchaseOrderApprovalsList

This is a read-only list available on an update and maps to the History/Approvals tab in the UI.

Accessing Serial or Lot Number Data for Line Items

As of the 2011.2 endpoint, code to access serial number or lot number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the inventory detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Linking Purchase Orders to a Vendor Bill

The 2012.2 endpoint (and any endpoint to come after) supports transaction links between purchase orders and vendor bills, allowing you to populate a vendor bill with the items and expenses from one or more purchase orders. Two approaches are supported for linking purchase order(s) to a vendor bill.

First Approach

The simpler approach is to use the purchaseOrderList in a vendor bill add request. When you include purchase order internalId or externalId value(s) in the purchaseOrderList, all of the items and expenses from the specified purchase order(s) are added to the vendor bill's itemList and expenseList. Use this approach when you want a vendor bill to contain all of the line items from linked purchase orders. See SOAP Request (Sample 1 - Approach 1) for details.

Important: If you set values for a vendor bill's purchaseOrderList in a request, you cannot set values for its itemList or expenseList in the same request.

Second Approach

The second approach is to send a vendor bill initialize request that references one or more purchase orders by id. The initializeResponse returns itemList and expenseList data from the referenced purchase order(s). This data includes orderDoc and orderLine field values that together provide a unique identifier for each line item. You can then send a vendor bill add request setting values for the vendor bill's itemList and expenseList with data copied from the response. You should copy only the data from the specific purchase order items and expenses that you want to be included in the vendor bill. This approach lets you select a subset of line items from each linked purchase order, rather than including all linked purchase order line items in a vendor bill.

In the second approach, you will use the initialize operation, which supports multiple purchase orders per vendor. Also, the orderDoc and orderLine fields are included in the vendor bill itemList and expenseList. For details, see SOAP Request (Sample 2 - Approach 2) and the accompanying SOAP Response and Java.

Important: If you reference multiple purchase orders with either of these approaches, all referenced purchase orders must be for the same vendor.

Code Samples

SOAP Request (Sample 1 - Approach 1)

Using an add request to populate a vendor bill with the items and expenses from multiple purchase orders:

```
  <soapenv:Body>
        <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
           <record xsi:type="ns6:VendorBill" xmlns:ns6="urn:purchases_2017_1.transactions.webs
ervices.netsuite.com">
              <ns6:entity internalId="98" xsi:type="ns7:RecordRef" xmlns:ns7="urn:core_2017_1.
platform.webservices.netsuite.com"/>
              <ns6:purchaseOrderList xsi:type="ns8:RecordRefList" xmlns:ns8="urn:core_2017_1.p
latform.webservices.netsuite.com">
                 <ns8:recordRef internalId="2659" type="purchaseOrder" xsi:type="ns8:RecordRef
"/>
                 <ns8:recordRef internalId="2661" type="purchaseOrder" xsi:type="ns8:RecordRef
"/>
              </ns6:purchaseOrderList>
           </record>
        </add>
     </soapenv:Body>
```

SOAP Request (Sample 2 - Approach 2)

Using an initialize request to populate a vendor bill with the items and expenses from multiple purchase orders. This approach lets you select a subset of line items from each linked purchase order, rather than including all linked purchase order line items in a vendor bill.

```
 <soapenv:Body>
        <initialize xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
           <initializeRecord>
              <ns7:type xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">vendorBi
ll</ns7:type>
              <ns8:referenceList xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com"
>
                 <ns8:initializeRef internalId="2659" type="purchaseOrder"/>
                 <ns8:initializeRef internalId="2661" type="purchaseOrder"/>
              </ns8:referenceList>
           </initializeRecord>
        </initialize>
     </soapenv:Body>
```

SOAP Response (Sample 2)

```
          <initializeResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
             <readResponse>
                <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
                <record xsi:type="tranPurch:VendorBill" xmlns:tranPurch="urn:purchases_2017_1.tr
ansactions.webservices.netsuite.com">
                   <tranPurch:entity internalId="98" xmlns:platformCore="urn:core_2017_1.platfor
m.webservices.netsuite.com">
                      <platformCore:name>Test Vendor</platformCore:name>
                   </tranPurch:entity>
                   <tranPurch:subsidiary internalId="1" xmlns:platformCore="urn:core_2017_1.plat
form.webservices.netsuite.com">
                      <platformCore:name>Parent Company</platformCore:name>
                   </tranPurch:subsidiary>
                   <tranPurch:approvalStatus internalId="2" xmlns:platformCore="urn:core_2017_1.
platform.webservices.netsuite.com">
                      <platformCore:name>Approved</platformCore:name>
                   </tranPurch:approvalStatus>
                   <tranPurch:postingPeriod internalId="126" xmlns:platformCore="urn:core_2017_1
.platform.webservices.netsuite.com">
                      <platformCore:name>Apr 2012</platformCore:name>
                   </tranPurch:postingPeriod>
                   <tranPurch:tranDate>2012-04-30T00:00:00.000-07:00</tranPurch:tranDate>
                   <tranPurch:currencyName>US USD</tranPurch:currencyName>
                   <tranPurch:exchangeRate>1.0</tranPurch:exchangeRate>
                   <tranPurch:dueDate>2012-04-30T00:00:00.000-07:00</tranPurch:dueDate>
                   <tranPurch:userTotal>121.0</tranPurch:userTotal>
                   <tranPurch:currency internalId="1" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com">
                      <platformCore:name>US USD</platformCore:name>
                   </tranPurch:currency>
                   <tranPurch:expenseList>
                      <tranPurch:expense>
                         <tranPurch:orderDoc>2661</tranPurch:orderDoc>
                         <tranPurch:orderLine>1</tranPurch:orderLine>
                         <tranPurch:category internalId="2" xmlns:platformCore="urn:core_2017_1.
platform.webservices.netsuite.com">
                            <platformCore:name>Entertainment</platformCore:name>
                         </tranPurch:category>
                         <tranPurch:account internalId="59" xmlns:platformCore="urn:core_2017_1.
platform.webservices.netsuite.com">
                            <platformCore:name>Advertising</platformCore:name>
                         </tranPurch:account>
                         <tranPurch:amount>2.0</tranPurch:amount>
                         <tranPurch:isBillable>false</tranPurch:isBillable>
                      </tranPurch:expense>
                      <tranPurch:expense>
                         <tranPurch:orderDoc>2659</tranPurch:orderDoc>
                         <tranPurch:orderLine>1</tranPurch:orderLine>
                         <tranPurch:category internalId="2" xmlns:platformCore="urn:core_2017_1.
platform.webservices.netsuite.com">
                            <platformCore:name>Entertainment</platformCore:name>
                         </tranPurch:category>
```

```
                        <tranPurch:account internalId="59" xmlns:platformCore="urn:core_2017_1.
platform.webservices.netsuite.com">
                           <platformCore:name>Advertising</platformCore:name>
                        </tranPurch:account>
                        <tranPurch:amount>5.0</tranPurch:amount>
                        <tranPurch:isBillable>false</tranPurch:isBillable>
                     </tranPurch:expense>
                     <tranPurch:expense>
                        <tranPurch:orderDoc>2661</tranPurch:orderDoc>
                        <tranPurch:orderLine>2</tranPurch:orderLine>
                        <tranPurch:category internalId="1" xmlns:platformCore="urn:core_2017_1.
platform.webservices.netsuite.com">
                           <platformCore:name>Travel</platformCore:name>
                        </tranPurch:category>
                        <tranPurch:account internalId="59" xmlns:platformCore="urn:core_2017_1.
platform.webservices.netsuite.com">
                           <platformCore:name>Advertising</platformCore:name>
                        </tranPurch:account>
                        <tranPurch:amount>10.0</tranPurch:amount>
                        <tranPurch:isBillable>false</tranPurch:isBillable>
                     </tranPurch:expense>
                     <tranPurch:expense>
                        <tranPurch:orderDoc>2659</tranPurch:orderDoc>
                        <tranPurch:orderLine>2</tranPurch:orderLine>
                        <tranPurch:category internalId="1" xmlns:platformCore="urn:core_2017_1.
platform.webservices.netsuite.com">
                           <platformCore:name>Travel</platformCore:name>
                        </tranPurch:category>
                        <tranPurch:account internalId="59" xmlns:platformCore="urn:core_2017_1.
platform.webservices.netsuite.com">
                           <platformCore:name>Advertising</platformCore:name>
                        </tranPurch:account>
                        <tranPurch:amount>100.0</tranPurch:amount>
                        <tranPurch:isBillable>false</tranPurch:isBillable>
                     </tranPurch:expense>
                  </tranPurch:expenseList>
                  <tranPurch:itemList>
                     <tranPurch:item>
                        <tranPurch:item internalId="39" xmlns:platformCore="urn:core_2017_1.pla
tform.webservices.netsuite.com">
                           <platformCore:name>Item No Tax 2 Lbs(All)</platformCore:name>
                        </tranPurch:item>
                        <tranPurch:orderDoc>2659</tranPurch:orderDoc>
                        <tranPurch:orderLine>3</tranPurch:orderLine>
                        <tranPurch:quantity>1.0</tranPurch:quantity>
                        <tranPurch:description>Item No Tax 2 Lbs(All)</tranPurch:description>
                        <tranPurch:rate>1.00</tranPurch:rate>
                        <tranPurch:amount>1.0</tranPurch:amount>
                        <tranPurch:isBillable>false</tranPurch:isBillable>
                     </tranPurch:item>
                     <tranPurch:item>
                        <tranPurch:item internalId="38" xmlns:platformCore="urn:core_2017_1.pla
tform.webservices.netsuite.com">
                           <platformCore:name>Item No Tax 2 Lbs(US)</platformCore:name>
                        </tranPurch:item>
                        <tranPurch:orderDoc>2659</tranPurch:orderDoc>
                        <tranPurch:orderLine>4</tranPurch:orderLine>
                        <tranPurch:quantity>1.0</tranPurch:quantity>
                        <tranPurch:rate>1.00</tranPurch:rate>
                        <tranPurch:amount>1.0</tranPurch:amount>
                        <tranPurch:isBillable>false</tranPurch:isBillable>
                     </tranPurch:item>
                     <tranPurch:item>
                        <tranPurch:item internalId="36" xmlns:platformCore="urn:core_2017_1.pla
tform.webservices.netsuite.com">
                           <platformCore:name>Item Tax 2 Lbs(All)</platformCore:name>
                        </tranPurch:item>
                        <tranPurch:orderDoc>2661</tranPurch:orderDoc>
                        <tranPurch:orderLine>3</tranPurch:orderLine>
                        <tranPurch:quantity>1.0</tranPurch:quantity>
                        <tranPurch:rate>1.00</tranPurch:rate>
                        <tranPurch:amount>1.0</tranPurch:amount>
                        <tranPurch:isBillable>false</tranPurch:isBillable>
                     </tranPurch:item>
                     <tranPurch:item>
                        <tranPurch:item internalId="29" xmlns:platformCore="urn:core_2017_1.pla
tform.webservices.netsuite.com">
                           <platformCore:name>Item Tax 2 Lbs(US)</platformCore:name>
                        </tranPurch:item>
                        <tranPurch:orderDoc>2661</tranPurch:orderDoc>
                        <tranPurch:orderLine>4</tranPurch:orderLine>
                        <tranPurch:quantity>1.0</tranPurch:quantity>
                        <tranPurch:rate>1.00</tranPurch:rate>
                        <tranPurch:amount>1.0</tranPurch:amount>
                        <tranPurch:isBillable>false</tranPurch:isBillable>
                     </tranPurch:item>
                  </tranPurch:itemList>
                  <tranPurch:customFieldList xmlns:platformCore="urn:core_2017_1.platform.webse
rvices.netsuite.com">
                     <platformCore:customField internalId="custbody_633637_asubmit" xsi:type="p
latformCore:BooleanCustomFieldRef">
                        <platformCore:value>false</platformCore:value>
                     </platformCore:customField>
                     <platformCore:customField internalId="custbody_633637_bsubmit" xsi:type="p
latformCore:BooleanCustomFieldRef">
                        <platformCore:value>false</platformCore:value>
                     </platformCore:customField>
                     <platformCore:customField internalId="custbody_633637_bload" xsi:type="pla
tformCore:BooleanCustomFieldRef">
                        <platformCore:value>false</platformCore:value>
                     </platformCore:customField>
                  </tranPurch:customFieldList>
               </record>
            </readResponse>
         </initializeResponse>
```

Java (Sample 2)

```
public void initializeVendorBillByPurchaseOrders() throws Exception
```

```
    {
       this.login();

       InitializeRefList purchaseOrderList = new InitializeRefList();
       InitializeRef[] initializeRefArray = new InitializeRef[2];
       initializeRefArray[0] = new InitializeRef(null, InitializeRefType.purchaseOrder, "510", n
ull);
       initializeRefArray[1] = new InitializeRef(null, InitializeRefType.purchaseOrder, "511", n
ull);
       purchaseOrderList.setInitializeRef(initializeRefArray);
       InitializeRecord initializeRecord = new InitializeRecord(InitializeType.vendorBill, null,
 null, purchaseOrderList);
       ReadResponse response = _port.initialize(initializeRecord);
       VendorBill vendorBill = (VendorBill) response.getRecord();
    }
```

Requisition

You use the requisition record to initiate the purchase process for goods and services needed within your company.

This record is available only when the Requisitions feature is enabled, at Setup > Company > Enable Features, on the Transactions subtab.

This record type is defined in the purchases XSD, where it is called PurchaseRequisition.

In the user interface, you access this record at Transactions > Purchases > Enter Requisitions. For help working with this record in the user interface, see the help topic Entering a Requisition.

For more on using web services to interact with the requisition record, see the following:

- Requisition Supported Operations
- Requisition Body and Sublist Fields
- Requisition Code Samples

Requisition Supported Operations

The following operations are supported for use with the requisition record:

add | addList | attach / detach |delete | deleteList | get | getDeleted | getList | getSelectValue | initialize / initializeList | search | searchMore | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Note the following:

- The initialize function is used to transform a requisition record into a purchase order record. For an example, see Initializing a Purchase Order from a Requisition Record.
- To search for requisition records, use TransactionSearchBasic and set the type field to _requisition.

Requisition Body and Sublist Fields

Note the following:

- When you are adding a requisition using the standard entry form, the system automatically populates required body fields. For example, the requestor field is automatically populated with the user ID of the user sending the SOAP request.
- When adding a requisition, you much include at least one line in the itemList sublist.

Note: For full details about all body fields, sublist fields, search filters, and search joins available to this record, refer to the purchase requisition reference page in the SuiteTalk Schema Browser. For information on using the Schema Browser, see SuiteTalk Schema Browser.

Requisition Code Samples

Refer to the following examples for help integrating with the requisition record.

Adding a Requisition Record

The following C# sample shows how to add a requisition record.

```
private void addRequisition()
{

   PurchaseRequisition myRequisition = new PurchaseRequisition();

   myRequisition.externalId = "1050A";

   PurchaseRequisitionItemList myItemList = new PurchaseRequisitionItemList();
   myItemList.purchaseRequisitionItem = new PurchaseRequisitionItem[1];
   myRequisition.itemList = myItemList;

   RecordRef myFirstItem = new RecordRef();
   myFirstItem.type = RecordType.inventoryItem;
   myFirstItem.typeSpecified = true;
   myFirstItem.internalId = "959";

   myItemList.purchaseRequisitionItem[0] = new PurchaseRequisitionItem();
   myItemList.purchaseRequisitionItem[0].item = myFirstItem;
   myItemList.purchaseRequisitionItem[0].amount = 1;
   myItemList.purchaseRequisitionItem[0].amountSpecified = true;

   _service.add(myRequisition);

}
```

Initializing a Purchase Order from a Requisition Record

The following Java sample shows how to initialize a purchase order from a requisition record. In this example, assume that mrr is a method you have created to generate a RecordRef based on internal ID.

```
PurchaseOrder po = (PurchaseOrder) c.initialize(new InitializeRef(null, InitializeRefType.purch
aseRequisition, reqId, null), InitializeType.purchaseOrder);
po.setEntity(mrr("105"));
String poId = c.addRecord(po);
```

Return Authorization

A return authorization transaction, also known as a return materials authorization (RMA), records information about an expected return of items from a customer, including the item IDs, vendors, quantities, and prices (which determine the amounts to be credited or refunded to the customer. This transaction is non-posting. It is available when the Return Authorizations feature is enabled.

For more details about this type of transaction, see the help topic Customer Return Management.

The return authorization record is defined in the tranCust (customers) XSD.

Supported Operations

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's return authorization reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Working with Credit Card Data

When working with credit card data, be aware of the following security features:

- When adding a transaction that uses a credit card number, you cannot identify the credit card number using a masked value such as ************5151. You must enter the full 16-digit number, or you can identify an existing credit card record through a RecordRef. (You can identify the full number using the **ccNumber** field. You can reference an existing record using the **creditCard** field.)
- Searches do not work if they include the operator **is** or **isNot** in conjunction with the ccNumber field. The only search operators that can apply to this field are **empty** and **notEmpty**.

Initializing Return Authorizations

You can initialize a Return Authorization from a Cash Sale, an Invoice, or a Sales Order.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

Working with Return Authorization Sublists

The SuiteTalk Schema Browser includes all sublists associated with the return authorization record. See the following information for usage notes regarding specific return authorization sublists. Usage notes are not provided for every sublist type.

ReturnAuthorizationItemList

To access per line tax fields, the Per-Line Taxes on Transactions feature must be enabled.

Accessing Serial or Lot Number Data for Line Items

As of the 2011.2 endpoint, code to access serial number or lot number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the inventory detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Sales Order

A sales order transaction records a commitment to sell goods or services to a customer. Sales orders have no accounting impact until items are shipped or services are completed.

For details about this type of transaction, see the help topic Sales Orders.

The sales order record is defined in the tranSales (sales) XSD.

Supported Operations

The following operations can be used to modify sales order records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's sales order reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Working with Credit Card Data

When working with credit card data, be aware of the following security features:

- When adding a transaction that uses a credit card number, you cannot identify the credit card number using a masked value such as ************5151. You must enter the full 16-digit number, or you can identify an existing credit card record through a RecordRef. (You can identify the full number using the **ccNumber** field. You can reference an existing record using the **creditCard** field.)
- Searches do not work if they include the operator **is** or **isNot** in conjunction with the ccNumber field. The only search operators that can apply to this field are **empty** and **notEmpty**.

Initializing Sales Orders

You can initialize a sales order from an Estimate/Quote or an Opportunity.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

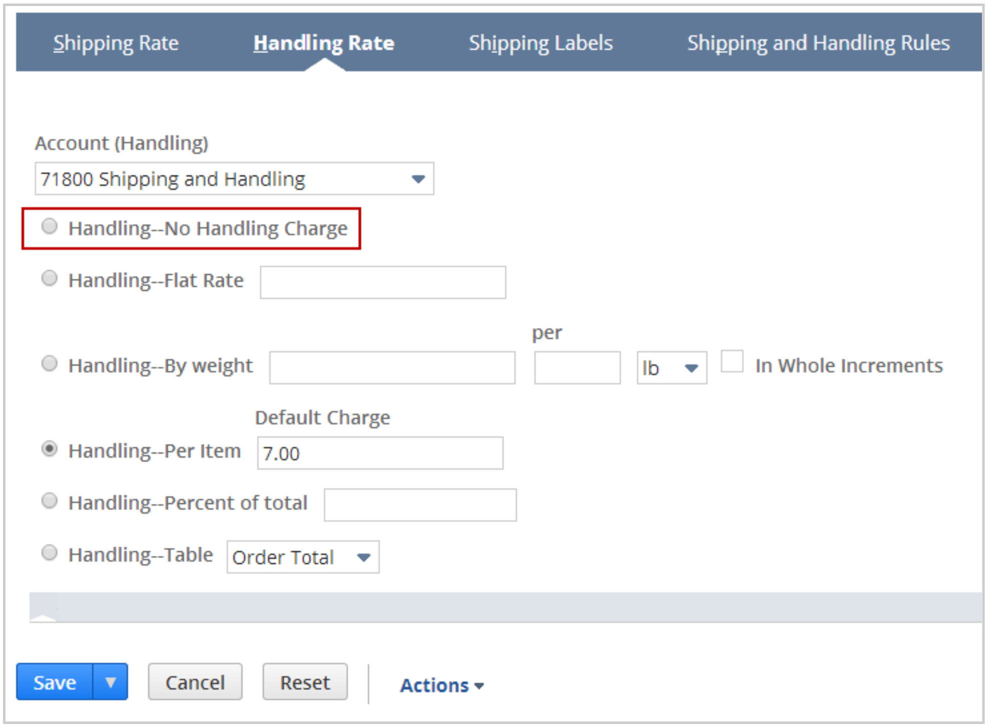
Setting Handling Cost

If you want to set a value for handlingCost, be aware of the following guidelines.

Only Certain Shipping Methods Allow Handling Cost To Be Set

In both the UI and in web services, the default behavior of the system is that you cannot specify a handling cost on a new sales order. To be able to enter a value for handling cost, you must choose a shipping method that enables the handling cost field.

You can identify an appropriate shipping method by reviewing the details for the corresponding shipping item. Choose Lists > Accounting > Shipping Items, then select the shipping item you want to review (each shipping item corresponds with a shipping method). In the item record, review the details on the Handling Rate subtab. As long as the Handling - No Handling Charge is **not** selected, then choosing this shipping item lets you set a handlingCost value.



Account (Handling): 71800 Shipping and Handling

- Handling--No Handling Charge
- Handling--Flat Rate
- Handling--By weight — per lb — In Whole Increments
- [✓ Handling--Per Item] — Default Charge: 7.00
- Handling--Percent of total
- Handling--Table — Order Total

During an add operation, you must set the shipMethod value in the same SOAP request where you set the handlingCost value. If you fail to set any value for shipMethod, or if you choose a value that does not enable the handling cost field, the system generates an error reading in part “You do not have permissions to set a value for element handlingcost.”

You Must Specify a Shipping Cost

If you are doing an add operation, be aware that in some cases the system may overwrite the handlingCost value specified in your SOAP request. To avoid this, make sure that your SOAP request include a shippingCost value. If you fail to include a shippingCost, the system overwrites your handlingCost value with an automatically generated value.

This behavior is specific to the add operation. During an update, it is possible to set a value for handlingCost without specifying the shippingCost.

Adding a Sales Order with an Item Sublist

C#

```
public virtual void  addSalesOrder()
      {
         // This operation requires a valid session
         this.login(true);

         SalesOrder so = new SalesOrder();
         // Set customer entity

         _out.writeLn("\nPlease enter the following customer information. "
          + "Note that some fields have already been populated. ");
         _out.write("  Customer entity name: ");
```

```
        CustomerSearch custSearch = new CustomerSearch();
        SearchStringField customerEntityID = new SearchStringField();
        customerEntityID.@operator = SearchStringFieldOperator.@is;
        customerEntityID.operatorSpecified = true;
        customerEntityID.searchValue = _out.readLn();

        CustomerSearchBasic custBasic = new CustomerSearchBasic();
        custBasic.entityId = customerEntityID;

        //custSearch.setEntityId(customerEntityID);

        custSearch.basic = custBasic;

        // Search for the customer entity
        SearchResult res = _service.search(custSearch);

        if (res.status.isSuccess)
        {
           if (res.recordList != null && res.recordList.Length == 1)
           {
              RecordRef customer = new RecordRef();
              customer.type = RecordType.customer;
              customer.typeSpecified = true;
              System.String entID = ((Customer) (res.recordList[0])).entityId;
              customer.name = entID;
              customer.internalId = ((Customer) (res.recordList[0])).internalId;
              so.entity = customer;

              // set the transaction date and status
              so.tranDate = new System.DateTime();
              so.orderStatus = SalesOrderOrderStatus._pendingApproval;

              // Enter the nsKey for inventory items to be added to the SO
              _out.writeLn("\nPlease enter the nsKey values for
               INVENTORY ITEMS separated by
              commas (do not enter discount or subtotal items).");
              System.String itemKeys = _out.readLn();
              System.String[] itemKeysArray = itemKeys.Split(new Char[] {','});

              ReadResponse[] readRes = getInventoryItemList(itemKeysArray);

              // Determine the number of valid inventory item nsKeys entered
              ArrayList vec = new ArrayList();
              for (int a = 0; a < readRes.Length; a++)
              {
                 if (readRes[a].record != null)
                 {
                    vec.Add(readRes[a].record);
                 }
              }
              SalesOrderItem[] salesOrderItemArray = new SalesOrderItem[vec.Count];

              // Create the correct sales order items and populate the
              // quantity
              for (int i = 0; i < vec.Count; i++)
```

```
                {
                   if (vec[i] is InventoryItem)
                   {
                      RecordRef item = new RecordRef();
                      item.type = RecordType.inventoryItem;
                      item.typeSpecified = true;
                      item.internalId = ((InventoryItem) (vec[i])).internalId;
                      salesOrderItemArray[i] = new SalesOrderItem();
                      salesOrderItemArray[i].item = item;

                      _out.writeLn("\nPlease enter quantity for " + ((InventoryItem)
                      (vec[i])).itemId);
                      System.Double quantity = System.Double.Parse(_out.readLn());
                      salesOrderItemArray[i].quantity = quantity;
                      salesOrderItemArray[i].quantitySpecified = true;
                   }
                }

                //SalesOrderItemList salesOrderItemList = new
                   SalesOrderItemList(salesOrderItemArray, true);
                //TODO: No constructor present for two argument.
                SalesOrderItemList salesOrderItemList = new SalesOrderItemList();
                salesOrderItemList.item = salesOrderItemArray;

                so.itemList = salesOrderItemList;

                WriteResponse writeRes = _service.add(so);
                if (writeRes.status.isSuccess)
                {
                   _out.writeLn("\nSales order created successfully");
                }
                else
                {
                   _out.error(getStatusDetails(writeRes.status));
                }
             }
             else
             {
                _out.writeLn("\nSales order is not created because 0 or more than 1
                   customer records found for the entityID given");
             }
          }
          else
          {
             _out.error(getStatusDetails(res.status));
          }
       }
```

Java

```
public void addSalesOrder() throws RemoteException,
         ExceededUsageLimitFault, UnexpectedErrorFault, InvalidSessionFault,
         ExceededRecordCountFault {
      // This operation requires a valid session
      this.login(true);

      SalesOrder so = new SalesOrder();
      // Set customer entity

      _console.writeLn("\nPlease enter the following customer information. "
            + "Note that some fields have already been populated. ");
      _console.write("  Customer entity name: ");

      CustomerSearch custSearch = new CustomerSearch();
      SearchStringField customerEntityID = new SearchStringField();
      customerEntityID.setOperator(SearchStringFieldOperator.is);
      customerEntityID.setSearchValue(_console.readLn());

      CustomerSearchBasic custBasic = new CustomerSearchBasic();
      custBasic.setEntityId(customerEntityID);

      //custSearch.setEntityId(customerEntityID);

      custSearch.setBasic(custBasic);

      // Search for the customer entity
      SearchResult res = _port.search(custSearch);

      if (res.getStatus().isIsSuccess()) {
         if (res.getRecordList().getRecord()!=null && res.getRecordList().getRecord().length ==
1) {
            RecordRef customer = new RecordRef();
            customer.setType(RecordType.customer);
            String entID = ((Customer) (res.getRecordList().getRecord(0)))
                  .getEntityId();
            customer.setName(entID);
            customer.setInternalId(((Customer) (res.getRecordList()
                  .getRecord(0))).getInternalId());
            so.setEntity(customer);

            // set the transaction date and status
            so.setTranDate(Calendar.getInstance());
            so.setOrderStatus(SalesOrderOrderStatus._pendingApproval);

            // Enter the nsKey for inventory items to be added to the SO
            _console
                  .write("\nPlease enter the Internal Ids values for INVENTORY ITEMS separated
            by commas (do not enter discount or subtotal items): ");
            String itemKeys = _console.readLn();
            String[] itemKeysArray = itemKeys.split(",");

            ReadResponse[] readRes = getInventoryItemList(itemKeysArray);

            // Determine the number of valid inventory item nsKeys entered
            Vector vec = new Vector();
            for (int a = 0; a < readRes.length; a++) {
               if (readRes[a].getRecord() != null) {
                  vec.add(readRes[a].getRecord());
               }
            }
            SalesOrderItem[] salesOrderItemArray = new SalesOrderItem[vec
                  .size()];

            // Create the correct sales order items and populate the
            // quantity
            for (int i = 0; i < vec.size(); i++) {
               if (vec.get(i) instanceof InventoryItem) {
                  RecordRef item = new RecordRef();
                  item.setType(RecordType.inventoryItem);
                  item.setInternalId(((InventoryItem) (vec.get(i)))
                        .getInternalId());
                  salesOrderItemArray[i] = new SalesOrderItem();
                  salesOrderItemArray[i].setItem(item);

                  _console.write("\nPlease enter quantity for "
                        + ((InventoryItem) (vec.get(i))).getItemId() + ": ");
                  Double quantity = Double.valueOf(_console.readLn());
                  salesOrderItemArray[i].setQuantity(quantity);
               }
            }
            SalesOrderItemList salesOrderItemList = new SalesOrderItemList(
                  salesOrderItemArray, true);
            so.setItemList(salesOrderItemList);

            WriteResponse writeRes = _port.add(so);
            if (writeRes.getStatus().isIsSuccess()) {
               _console.writeLn("\nSales order created successfully");
            } else {
               _console.error(getStatusDetails(writeRes.getStatus()));
            }
         } else {
            _console
                  .writeLn("\nSales order is not created because 0 or more than 1 customer
                  records found for the entityID given");
         }
      } else {
         _console.error(getStatusDetails(res.getStatus()));
      }

   }
```

Setting the Shipping Address on a Sales Order

The following code illustrates how to set the shipTo address on a sales order.

SOAP Request

```
<soapenv:Body>
<add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<record xsi:type="ns6:SalesOrder" xmlns:ns6="urn:sales_2017_1.transactions.webservices.netsuite
.com">
   <ns6:entity internalId="103 customer" xsi:type="ns7:RecordRef"
      xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">
      <ns7:name xsi:type="xsd:string">Vintrust</ns7:name>
   </ns6:entity>
   <ns6:tranDate xsi:type="xsd:dateTime">2006-03-07T18:16:44.000Z</ns6:tranDate>
   <ns6:orderStatus xsi:type="ns8:SalesOrderOrderStatus"
      xmlns:ns8="urn:types.sales_2017_1.transactions.webservices.netsuite.com">_pendingApproval

   </ns6:orderStatus>
   <ns6:shipAddressList internalId="84" xsi:type="ns9:RecordRef"
      xmlns:ns9="urn:core_2017_1.platform.webservices.netsuite.com"/>
   <ns6:itemList replaceAll="true" xsi:type="ns6:SalesOrderItemList">
      <ns6:item xsi:type="ns6:SalesOrderItem">
         <ns6:item internalId="18 inventoryItem" xsi:type="ns10:RecordRef"
            xmlns:ns10="urn:core_2017_1.platform.webservices.netsuite.com"/>
         <ns6:quantity xsi:type="xsd:double">2.0</ns6:quantity>
      </ns6:item>
   </ns6:itemList>
</record>
</add>
</soapenv:Body>
```

C#

```
Customer customer = // some customer
CustomerAddressbook shipTo = // some customer address book entry
RecordRef rr = new RecordRef();
rr.setInternalId(shipTo.getInternalId());
rr.setName("addressBook");
salesOrder.setShipAddressList(rr);
```

Java

```
//block setting shipping address to something other than default
RecordRef altShipAddress = new RecordRef();
altShipAddress.setInternalId("84");
so.setShipAddressList(altShipAddress);
//end block
WriteResponse writeRes = _port.add(so);
```

Working with Sales Order Sublists

The SuiteTalk Schema Browser includes all sublists associated with the sale order record. See the following information for usage notes regarding specific sales order sublists. Note that usage notes are not provided for every sublist type.

SalesOrderSalesTeamList

This list is only available when the Team Selling feature is enabled.

Calculating Shipping Cost for Sales Orders

Web services calculation of shipping cost for sales orders varies slightly from the UI behavior. Because the sales order includes a Calculate Shipping button in the UI, updates to item quantity or other values that may impact shipping cost do not automatically recalculate that field's value in the UI. In web services, updates to item quantity or other values that impact shipping cost always result in recalculation of altshippingcost. If your account is set up to automatically charge for shipping, this defaults to the value that corresponds to the shipping method selected. To prevent this recalculation, resubmit the value for altshippingcost whenever you update item quantity or other fields that impact shipping cost.

Note that this web services behavior applies when the Multiple Shipping Routes feature is not enabled. For details about web services behavior with this feature enabled, see Multiple Shipping Routes in Web Services.

Sales Order Payment Terms and Payment Methods

The NetSuite UI validates payment method values against payment terms values on sales order records, preventing values that do not match. In endpoints prior to 2010.1, web services allows the saving of both fields' values, even if they do not match. This behavior may result in unexpected errors on payments. It is recommended that you do not attempt to set payment method and terms at the same time for endpoints prior to 2010.1.

Associating Sales Orders with Opportunities

You can associate estimates, cash sales, sales orders, and invoices with opportunities. After a transaction other than an estimate is associated with an opportunity, the opportunity's status is automatically set to **Closed Won**. After an opportunity's status is set to **Closed Won**, it is no longer available to be selected on other cash sale, sales order, or invoice records.

Accessing Serial or Lot Number Data for Line Items

As of the 2011.2 endpoint, code to access serial number or lot number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the inventory detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

CyberSource Decision Manager Fields for Web Store Orders

As of the 2012.1 endpoint, sales order records include the following fields to be used in conjunction with CyberSource Decision Manager for third party Web store orders.

- shopperipaddress - captures the shopper's IP address, passes it to a newly created or updated sales order, and sends it to CyberSource with the authorization request.

Important: The shopperipaddress field is required for Web store orders made through web services. A web services order that does not include a value for this field is treated as an internal (MOTO) order.

- saveonauthdecline - boolean that indicates whether to save Web store orders that receive a credit card authorization decline.

- When value is set to True, declined order is saved and placed on Payment Hold for further review. (This is the default.)
- When value is set to False, declined order is not saved, and the shopper can retry submitting the order with a different payment method.

The following code illustrates how to add a Web store order when CyberSource Decision Manager is in use:

SOAP Request

```
<add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record externalId="Melon16973236" xsi:type="ns6:SalesOrder" xmlns:ns6="urn:sales_2
017_1.transactions.webservices.netsuite.com">
               <ns6:customForm internalId="109" xsi:type="ns7:RecordRef" xmlns:ns7="urn:core_20
17_1.platform.webservices.netsuite.com"/>
               <ns6:entity internalId="102" xsi:type="ns8:RecordRef" xmlns:ns8="urn:core_2017_1
.platform.webservices.netsuite.com"/>
               <ns6:paymentMethod internalId="8" type="paymentMethod" xsi:type="ns9:RecordRef"
xmlns:ns9="urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns6:shopperIpAddress xsi:type="xsd:string">167.216.129.12</ns6:shopperIpAddress
>
               <ns6:saveOnAuthDecline xsi:type="xsd:boolean">true</ns6:saveOnAuthDecline>
               <ns6:ccNumber xsi:type="xsd:string">4111111111111111</ns6:ccNumber>
               <ns6:ccExpireDate xsi:type="xsd:dateTime">2012-02-02T19:41:13.702Z</ns6:ccExpire
Date>
               <ns6:ccName xsi:type="xsd:string">James James</ns6:ccName>
               <ns6:creditCardProcessor internalId="4" xsi:type="ns10:RecordRef" xmlns:ns10="ur
n:core_2017_1.platform.webservices.netsuite.com"/>
               <ns6:getAuth xsi:type="xsd:boolean">true</ns6:getAuth>
               <ns6:itemList replaceAll="false" xsi:type="ns6:SalesOrderItemList">
                  <ns6:item xsi:type="ns6:SalesOrderItem">
                     <ns6:item internalId="447" xsi:type="ns11:RecordRef" xmlns:ns11="urn:core_
2017_1.platform.webservices.netsuite.com"/>
                     <ns6:quantity xsi:type="xsd:double">2.0</ns6:quantity>
                  </ns6:item>
                  <ns6:item xsi:type="ns6:SalesOrderItem">
                     <ns6:item internalId="451" xsi:type="ns12:RecordRef" xmlns:ns12="urn:core_
2017_1.platform.webservices.netsuite.com"/>
                     <ns6:quantity xsi:type="xsd:double">2.0</ns6:quantity>
                  </ns6:item>
               </ns6:itemList>
            </record>
         </add>
```

For more information about using CyberSource Decision Manager, see the help topic CyberSource. Specificall, read the help topic Set Up Decision Manager for Web Services Orders.

Statistical Journal Entry

The statistical journal entry record lets you increase or reduce the balance of statistical accounts.

This record is defined in the tranGeneral XSD, where it is called StatisticalJournalEntry.

To use this record, the Statistical Accounts feature must be enabled at Setup > Enable Features, on the Accounting subtab. Also, you must have already created at least one statistical account.

In the user interface, you access this record at Transactions > Financial > Make Statistical Journal Entries. For help working with this record in the user interface, see the help topic Making Statistical Journal Entries.

For more details on using SuiteTalk to work with the statistical journal entry record, see the following sections:

- Statistical Journal Entry Supported Operations
- Statistical Journal Entry Body and Sublist Fields
- Statistical Journal Entry Code Samples
- Common Errors with Statistical Journal Entries

Statistical Journal Entry Supported Operations

The following operations are supported for use with the statistical journal entry record:

add | addList | delete | deleteList | get | getList | search | searchMore | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Note: You cannot update the subsidiary or the unitsType body fields through the update operation. You can, however, update the subsidiary field through the user interface. To do this, delete the lines from the sublist, select a new subsidiary, add new lines, and then save the modified record.

Statistical Journal Entry Body and Sublist Fields

When you create a statistical journal entry, you must define certain body fields, and you must create at least one sublist line.

The required body fields are described in the following table.

Field	Description
unitsType	This field must reference an existing field listed at Lists > Accounting > Units of Measure. Later, when you populate the sublist, the account associated with each line must be of this unit type. This field cannot be updated after the statistical journal entry record has been created.
subsidiary (OneWorld only)	This field must reference an existing subsidiary listed at Company > Setup > Subsidiaries. Later, when you populate the sublist, each account you choose must be a valid account for this subsidiary.

Field	Description
	This field cannot be updated through web services after the statistical journal entry record has been created. You can update this field through the user interface. To do this, delete the lines from the sublist, select a new subsidiary, add new lines, and then save the modified record.

Note: With the 2015.2 and later WSDLs, you can also interact with the Memo body field. By contrast, the Memo field from the Lines sublist is available in all supported WSDLs.

Each sublist line must include values for the fields described in the following table.

Field	Description
account	This field must reference an existing statistical account listed at Lists > Accounting > Accounts. The account you choose must be valid for the subsidiary body field value, and it must use the same unit of measure specified in the unitsType body field.
debit	This field takes a double value that represents the adjustment you want to make to the account. Though the field is called debit, entering a positive amount increases the value of the account. Entering a negative amount decreases the value. This field is labeled “Amount” in the user interface.

Note: The unitLabel field is read-only. It is populated automatically when you set the unitsType field. (The unitlabel value is the unitstype’s base unit value. This value is defined in the corresponding unit of measure record.)

Field Definitions

For more details on body and sublist fields, see the SuiteTalk Schema Browser’s statistical journal entry reference page. The Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

The maximum number of lines per statistical journal entry is **1000** when using synchronous web services processing. With asynchronous web services processing, the limit is 10,000 lines. If your import includes transactions with more than 1,000 lines, CSV import can also serve as an alternative. For transactions submitted through CSV import, the limit is 10,000 lines per transaction.

Statistical Journal Entry Code Samples

For examples of how to work with statistical journal entries, refer to the following sections:

- Adding a Statistical Journal Entry Record
- Searching for Statistical Journal Entry Records

Adding a Statistical Journal Entry Record

The following example shows how to create a statistical journal entry record with three lines.

C#

```
private void addStatisticalJournalEntry()
{

   // Create the statistical journal entry object.

   StatisticalJournalEntry newSJE = new StatisticalJournalEntry();


   // Create an external ID.

   newSJE.externalId = "101A";


   // Define the subsidiary (OneWorld accounts only).

   RecordRef mySubsidiary = new RecordRef();
   mySubsidiary.type = RecordType.subsidiary;
   mySubsidiary.internalId = "1";
   newSJE.subsidiary = mySubsidiary;


   // Declare the unit of measure type. All accounts named in the sublist
   // must be accounts that use this measure type.

   RecordRef unitOfMeasureReference = new RecordRef();
   unitOfMeasureReference.type = RecordType.unitsType;
   unitOfMeasureReference.internalId = "4";
   newSJE.unitsType = unitOfMeasureReference;


   // Create a sublist that can have up to three lines.

   newSJE.lineList = new StatisticalJournalEntryLineList();
   newSJE.lineList.statisticalJournalEntryLine = new StatisticalJournalEntryLine[3];


   // Create RecordRefs to define the three accounts to be adjusted. These accounts
   // must all have a unit of measure that matches the one you referenced for the
   // record via the unitsType body field.

   RecordRef firstAccount = new RecordRef();
   firstAccount.internalId = "248";

   RecordRef secondAccount = new RecordRef();
   secondAccount.internalId = "249";

   RecordRef thirdAccount = new RecordRef();
   thirdAccount.internalId = "253";
   // Define the three adjustments. The first line below reduces the balance of an
   // account; the next two increase the balances of their accounts.

   newSJE.lineList.statisticalJournalEntryLine[0] = new StatisticalJournalEntryLine();
   newSJE.lineList.statisticalJournalEntryLine[0].account = firstAccount;
   newSJE.lineList.statisticalJournalEntryLine[0].debit = -500;
   newSJE.lineList.statisticalJournalEntryLine[0].debitSpecified = true;

   newSJE.lineList.statisticalJournalEntryLine[1] = new StatisticalJournalEntryLine();
   newSJE.lineList.statisticalJournalEntryLine[1].account = secondAccount;
   newSJE.lineList.statisticalJournalEntryLine[1].debit = 1200;
   newSJE.lineList.statisticalJournalEntryLine[1].debitSpecified = true;

   newSJE.lineList.statisticalJournalEntryLine[2] = new StatisticalJournalEntryLine();
   newSJE.lineList.statisticalJournalEntryLine[2].account = thirdAccount;
   newSJE.lineList.statisticalJournalEntryLine[2].debit = 2000;
   newSJE.lineList.statisticalJournalEntryLine[2].debitSpecified = true;

   _service.add(newSJE);

}
```

SOAP Request

```
<soap:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record xmlns:q1="urn:general_2017_1.transactions.webservices.netsuite.com" xsi:type="q1:
StatisticalJournalEntry" externalId="101A">
         <q1:subsidiary internalId="1" />
         <q1:unitsType internalId="4" />
         <q1:lineList>
            <q1:statisticalJournalEntryLine>
               <q1:account internalId="248" />
               <q1:debit>-500</q1:debit>
            </q1:statisticalJournalEntryLine>
            <q1:statisticalJournalEntryLine>
               <q1:account internalId="249" />
               <q1:debit>1200</q1:debit>
            </q1:statisticalJournalEntryLine>
            <q1:statisticalJournalEntryLine>
               <q1:account internalId="253" />
               <q1:debit>2000</q1:debit>
            </q1:statisticalJournalEntryLine>
         </q1:lineList>
      </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 internalId="7331" externalId="101A" type="statisticalJournalEntry" xsi:type="
platformCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/
>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Searching for Statistical Journal Entry Records

To search for a statistical journal entry record, use any of the following: TransactionSearchBasic, TransactionSearch, or TransactionSearchAdvanced. Use the type field to restrict the search to journal entries. Set the statistical field to true to further narrow the search to statistical journal entries. The following example would return all statistical journal entry records that exist in the account.

C#

```

private void searchStatisticalJournalEntry()
{

   // Create the search object.

   TransactionSearchBasic myStatisticalJournalEntrySearch = new TransactionSearchBasic();


   // Restrict the search should to journal entry records.

   myStatisticalJournalEntrySearch.type = new SearchEnumMultiSelectField();
   myStatisticalJournalEntrySearch.type.@operator = SearchEnumMultiSelectFieldOperator.anyOf;
   myStatisticalJournalEntrySearch.type.operatorSpecified = true;
   myStatisticalJournalEntrySearch.type.searchValue = new string[1];
   myStatisticalJournalEntrySearch.type.searchValue[0] = "_journal";


   // Restrict the search to statistical journal entry records.

   myStatisticalJournalEntrySearch.statistical = new SearchBooleanField();
   myStatisticalJournalEntrySearch.statistical.searchValue = true;
   myStatisticalJournalEntrySearch.statistical.searchValueSpecified = true;


   // Execute the search.

   SearchResult searchResult = _service.search(myStatisticalJournalEntrySearch);

}
```

SOAP Request

```
<soap:Body>
```

```
   <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <searchRecord xmlns:q1="urn:common_2017_1.platform.webservices.netsuite.com" xsi:type="q1
:TransactionSearchBasic">
         <q1:statistical>
            <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">true</search
Value>
         </q1:statistical>
         <q1:type operator="anyOf">
            <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">_journal</se
archValue>
         </q1:type>
      </searchRecord>
   </search>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
   <searchResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <platformCore:searchResult xmlns:platformCore="urn:core_2017_1.platform.webservices.netsu
ite.com">
         <platformCore:status isSuccess="true"/>
         <platformCore:totalRecords>9</platformCore:totalRecords>
         <platformCore:pageSize>1000</platformCore:pageSize>
         <platformCore:totalPages>1</platformCore:totalPages>
         <platformCore:pageIndex>1</platformCore:pageIndex>
         <platformCore:searchId>WEBSERVICES_1013519_022020141337158990178161l728_80125163d91</p
latformCore:searchId>
         <platformCore:recordList>
            <platformCore:record internalId="6713" xsi:type="tranGeneral:JournalEntry" xmlns:tr
anGeneral="urn:general_2017_1.transactions.webservices.netsuite.com">
               <tranGeneral:postingPeriod internalId="315">
                <platformCore:name>Jan 2014</platformCore:name>
               </tranGeneral:postingPeriod>
               <tranGeneral:tranDate>2014-01-20T15:00:00.000-08:00</tranGeneral:tranDate>
               <tranGeneral:currency internalId="1">
                  <platformCore:name>US Dollar</platformCore:name>
               </tranGeneral:currency>
               <tranGeneral:exchangeRate>1.0</tranGeneral:exchangeRate>
               <tranGeneral:tranId>GJ10051</tranGeneral:tranId>
               <tranGeneral:subsidiary internalId="1">
                  <platformCore:name>Parent Company</platformCore:name>
               </tranGeneral:subsidiary>
               <tranGeneral:approved>false</tranGeneral:approved>
               <tranGeneral:createdDate>2014-01-21T09:09:45.000-08:00</tranGeneral:createdDate>

               <tranGeneral:lastModifiedDate>2014-02-11T08:21:16.000-08:00</tranGeneral:lastMod
ifiedDate>
               <tranGeneral:customFieldList>
                  <platformCore:customField internalId="562" scriptId="custbody_adjustment_jour
nal" xsi:type="platformCore:BooleanCustomFieldRef">
                     <platformCore:value>false</platformCore:value>
                  </platformCore:customField>
                  <platformCore:customField internalId="505" scriptId="custbody_report_timestam
```

```
p" xsi:type="platformCore:StringCustomFieldRef">
                     <platformCore:value>21/01/2014 09:08:41</platformCore:value>
                  </platformCore:customField>
            </tranGeneral:customFieldList>
               </platformCore:record>
            ...
         </platformCore:recordList>
      </platformCore:searchResult>
 </searchResponse>
</soapenv:Body>
```

Common Errors with Statistical Journal Entries

When you are working with statistical journal entry records, the following failure notices can occur in your SOAP responses.

Invalid account reference key x for subsidiary y

This error signifies a problem with the account record you referenced in one of your lines. The message can indicate either of the following problems:

- The account record identifies a subsidiary different from the one you defined for your statistical journal entry record through the subsidiary body field.
- The account record indicates a unit of measure different from the one you defined for your statistical journal entry record through the unitsType body field. All lines of the record must use this same unit of measure.

Invalid subsidiary reference x, Cannot change Subsidiary

These errors signify that you attempted to modify the subsidiary value during an update operation. In web services, the subsidiary field can be set only when creating the record.

You must enter at least one line item for this transaction

This error indicates that you attempted to create a statistical journal entry record with no lines, or that you tried to update a record by removing all its lines. Every statistical journal entry record must have at least one line.

Time Bill (Track Time)

A time transaction, also known as TimeBill, records the hours worked by an employee. This transaction can be used to record billable hours and invoice customers. This transaction is available when the Time Tracking feature is enabled.

For details about this type of transaction, Managing Time Tracking.

The TimeBill record is defined in the tranEmp (employees) XSD.

Important: The TimeBill record is labelled as the Track Time record in the UI.

Supported Operations

The following operations can be used with TimeBill records:

add | addList | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's time bill reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Weekly Timesheets

The Weekly Timesheets feature works in conjunction with the Time Tracking feature. For more information on the Weekly Timesheets feature, see the help topic Using Weekly Timesheets.

The TimeBill record used for the Time Tracking feature is supported when the Weekly Timesheets feature is enabled. Each TimeBill instance is a standalone record that belongs to only one TimeSheet instance. The TimeBill record's read-only RecordRef field, timeSheet references the TimeSheet to which each TimeBill belongs.

When the Weekly Timesheets feature is enabled, each TimeSheet references its related TimeBills in the timeitem sublist. A single TimeSheet can reference multiple TimeBills.

TimeBills are not available currently through the TimeSheet record in web services. To get all TimeBills for a TimeSheet, use TimeBill search.

The timeSheet field is available to TimeBill searches. You can also use joins for the TimeSheet and TimeBill searches.

Price Field

As of the 2011.2 endpoint, if you create a time bill record with a Price field set to -1, the value of the field is blank, rather than Custom.

Adding a Time Bill

The following code shows how to add a time bill.

C#

```
private void addTimeBill()
{
   Console.WriteLine("\nEnter Employee name: ");

   // Look for the employee
   EmployeeSearch empSearch = new EmployeeSearch();
   SearchStringField employeeEntityID = new SearchStringField();
   employeeEntityID.@operator = SearchStringFieldOperator.@is;
   employeeEntityID.operatorSpecified = true;
   employeeEntityID.searchValue = Console.ReadLine();

   EmployeeSearchBasic empBasic = new EmployeeSearchBasic();
   empBasic.entityId = employeeEntityID;

   empSearch.basic = empBasic;

   // Run the search
   SearchResult result = _service.search(empSearch);

   if (result.status.isSuccess)
   {
      Console.WriteLine("\nEmployees found: " + result.recordList.Length);

      if (result.recordList != null && result.recordList.Length == 1)
      {
         Employee emp = (Employee)result.recordList[0];

         Console.WriteLine("\nEmployeeID: " + emp.internalId);

         // Instantiate new blank time bill record
         TimeBill timeBill = new TimeBill();

         // Instantiate a record object
         timeBill.employee = new RecordRef();
         timeBill.employee.type = RecordType.employee;
         timeBill.employee.internalId = emp.internalId;

         // Set the date of the time bill to today
         timeBill.tranDate = DateTime.Today;
         Console.WriteLine("\nToday: {0}", timeBill.tranDate);

         // Get the amount of hours to add
         timeBill.hours = new Duration();
         Console.WriteLine("\nEnter hours: ");

         // Convert the string to a double
         timeBill.hours.timeSpan = Double.Parse(Console.ReadLine());
```

```
            // Add the record
            WriteResponse response = _service.add(timeBill);

            // Display the result of the operation
            if(response.status.isSuccess)
            {
               Console.WriteLine("Record added");
            }
            else
            {
               Console.WriteLine("Record was not added");
            }
         }
         else
         {
            Console.WriteLine("\nSorry - No such Employee");
         }
      }
   }
```

Transfer Order

The Transfer Order transaction is used to move inventory between locations when the Multi-Location Inventory (MLI) feature is enabled. Existing integrations with external warehouse management systems can leverage this transaction to manage data about inventory movement between locations.

Transfer orders can initialize item fulfillment and item receipt transactions. See Item Fulfillment and Item Receipt for details about these transactions.

The transfer order record is defined in the tranInvt (inventory) XSD.

Supported Operations

The following operations can be used to modify transfer order records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's transfer order reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Transfer Order Features and Preferences

The Transfer Order transaction is available when the Multi-Location Inventory feature is enabled. Note that the Pick, Pack and Ship feature can affect transfer order workflow.

The following accounting preferences, available at Setup > Accounting >Accounting Preferences, on the Order Management subtab, apply to transfer orders:

- Default Transfer Order Status indicates whether transfer orders require approval by default.
  - Pending Approval means that by default the status of someone with permission must approve the order before it is processed.
  - Pending Fulfillment means that by default transfer orders are sent directly to the fulfillment queue without requiring further approval.
- Use Item Cost as Transfer Cost affects how the Transfer Price from each transfer order is used on the related item receipt transaction.
  - When this option is enabled, the transfer price is used as a declared shipping value for reference only, such as for insurance or international shipping, and the item cost from the item record is used on the item receipt. Also, when this option is enabled and different items are fulfilled in different item fulfillments, you must receive the items separately. In this case, a separate item receipt is created for each item fulfillment.
  - When this option is disabled, the transfer price on the transfer order is used as the item cost on the item receipt. In this case, both the item fulfillment and the item receipt resulting from the transfer order have G/L impact. Also, when this option is disabled and different items are fulfilled in different item fulfillments, you can receive all items in the same item receipt.

Transfer Order Workflow

The following is a rough outline of transfer order workflow. For more details, see the help topic Inventory Transfer Orders.

1. A transfer order is entered to schedule the movement of items and its status is set to either Pending Approval or Pending Fulfillment.

   A transfer order can be approved by setting the status to Pending Fulfillment.

   Note: A transfer order initially can be saved without a source location value, for example, in a case where an item is needed at the destination location but the source location has not yet been identified. However, the source location value is required for an approved transfer order.

2. After its status is set to Pending fulfillment, the transfer order can be fulfilled from the source location.
   - An item receipt transaction is created.
   - Items are committed out of the source location's inventory.
   - The On Order quantity of items increases at the destination location.
   - Fulfilled items are removed from the On Hand count at the source location.

    - The value of items in transit are removed from the Inventory Asset account and added to the Inventory in Transit account for the source location.
    - The status of the transfer order is set to Pending Receipt, or if the Pick, Pack and Ship feature is enabled, to Shipped.
3. After the transfer order status has been set to Pending Receipt (or to Shipped),Transfer order items can be received at the destination location.
    - An item receipt transaction is created.
    - The items are added to the destination location's inventory and increase the On Hand count.
    - The items' value is added to the Inventory Asset account for the destination location.
    - The On Order quantity of the item in the destination location decreases.

Serialized and Lot Inventory Transfer Orders

Serial or lot numbered items can be included as transfer order line items. Note the following requirements:

- Serial or lot numbers are limited to 3800 characters.
- Supported separators for serial numbers are: space, comma, or line break.
- A serial number is required for every serialized line item.

  For example, if the quantity of a line item is 2, two serial numbers are required.
- Lot numbers must use the following format: <Lot #>(<Quantity>)

  For example, to indicate a quantity of 100 for an item with lot number AAA-3400, format would be: AAA-3400(100)

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the inventory detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Usage

In the Advanced Subscription Billing feature, usage is connected to a subscription line. Examples of usage include money, time, cellular data, internet data, etc. The usage record is defined in the sales XSD.

For details about working with the usage record in the user interface, see the help topic Working with Usage.

Usage Prerequisites

To work with the usage record in SuiteTalk, you must have the Advanced Subscription Billing feature enabled for your account.

To add a usage record, you must first add the following prerequisite records in the user interface:

- Subscription
- Subscription Line

Usage Supported Operations

add | addList | delete | deleteList | get | getDeleted | getList | search | searchMore | searchNext | searchMoreWithId | getSavedSearch |update | getSelectValue | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Usage Notes and Field Summary

The usage record does not support user-event scripts.

Body Fields

When creating a usage record, you are required to define the following body fields:

Field Name	Label in UI	Description
`usageSubscription`	Subscription	The subscription name. This is a RecordRef field.
`usageSubscriptionLine`	Subscription Line	The subscription line name. This is a RecordRef field.
`usageQuantity`	Quantity	The usage quantity number. This is a double field.
`usageDate`	Date	The usage date. This is a dateTime field.

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s usage reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Vendor Bill

The vendor bill transaction records payables as they arrive from vendors, allowing you to pay bills from the payables list as they are due, and providing an accurate picture of payables at all points of the billing cycle.

For details about this type of transaction, Vendor Bills.

The vendor bill record is defined in the tranPurch (purchases) XSD.

Supported Operations

The following operations can be used with vendor bill records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's vendor bill reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Initializing Vendor Bills

You can initialize a vendor bill from a Vendor or a Purchase Order.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

For information specific to linking multiple purchase orders to a single vendor bill, see Linking Purchase Orders to a Vendor Bill.

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the inventory detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Vendor Credit

A vendor credit transaction creates a credit, from a vendor, that can be applied to a payables account. For example, a vendor credit transaction may occur when items are returned to a vendor or when a discount is negotiated with a vendor.

For details about this type of transaction, see the help topic Vendor Credits.

The vendor credit record is defined in the tranPurch (purchases) XSD.

Supported Operations

The following operations can be used with vendor credit records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's vendor credit reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Initializing Vendor Credits

You can initialize a vendor credit from a Vendor, a Vendor Bill, or a Vendor Return Authorization.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record.

Note: Subsidiary is passed as an initialize parameter to the vendor credit record.

For more information about this operation, see the help topic initialize / initializeList.

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the inventory detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Vendor Payment

A vendor payment transaction posts to the general ledger as an expense and the amount of the payment is deducted from your accounts payable total. A vendor payment can be applied to one or more vendor bills. Vendor payments can help to track expenditures and total payables due.

For details about this type of transaction, see the help topic Vendor Payments.

The vendor payment record is defined in the tranPurch (purchases) XSD.

Supported Operations

The following operations can be used with vendor payment records:

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's vendor payment reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Initializing Vendor Payments

You can initialize a vendor payment from an Employee, a Vendor, or a Vendor Bill.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

Applying Vendor Payments to Bills

As of the 2011.1 endpoint and later, you can use {doc, line} on the VendorPaymentApply sublist to identify the vendor bill line to which a payment should be applied. You can use the line element to uniquely identify a sublist entry when the **doc** element is not sufficient. The line element prevents problems with record creation due to duplicate keys.

Use the initialize operation to obtain the doc and line values from the vendor bill.

Warning: As of the 2011.1 endpoint and later, if you do not provide a value for the line element for an entry to VendorPaymentApply, you may receive the error: “Ambiguous sublist reference; multiple matches exist for key value <doc>”. Earlier endpoints do not return this error.

Vendor Return Authorization

A vendor return authorization is a non-posting transaction that tracks a return to a vendor, including the items to be returned, their quantities, the approval status, the shipment status, and the amount refunded or credited from the vendor. This type of transaction is available when the Vendor Return Authorizations feature is enabled.

The vendor return process includes four steps: creating a vendor return authorization record, approving or canceling the authorization, shipping items authorized to be returned, and crediting an authorized vendor return. For more details, see the help topic Vendor Return Authorization Overview.

The vendor return authorization record is defined in the tranPurch (purchases) XSD.

Supported Operations

The following operations can be used with vendor return authorization records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s vendor return authorization reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Initializing Vendor Return Authorizations

You can initialize a vendor return authorization from a Vendor or a Vendor Bill.

The web services initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

Accessing Serial or Lot Number Data for Line Items

As of the 2011.2 endpoint, code to access serial number or lot number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the Inventory Detail record to access bin and numbered inventory fields and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Work Order

Work order transactions track the production of assembly items needed for stock or to fill orders. Work orders track the quantities of assemblies that need to be built and the quantities of components, or member items, needed to do so. This type of transaction is available when the Assembly Items and Work Orders features are enabled and is used when the Allow Purchase of Assembly Items accounting preference is not enabled.

Special order work orders track assemblies for a particular sale, and Production work orders track assemblies to increase stock. Both use the same work order form, but Production work orders do not link to a sales transaction. Production work orders are generated when the back ordered quantity of an assembly reaches its assigned build point. After the build point is reached, a work order is added in the Mass Create Work Orders queue. For more details, see the help topic Assembly Work Orders.

The work order record is defined in the tranInvt (inventory) XSD.

Supported Operations

The following operations can be used with work order records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's work order reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Accessing Serial/Lot or Bin Data for Line Items

As of the 2011.2 endpoint, code to access serial number, lot number, and bin number data varies according to whether the Advanced Bin Management / Numbered Inventory Management feature is enabled.

- If this feature is enabled, you must use the 2011.2 endpoint or later to access the Inventory Detail subrecord and the most up-to-date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.
- If this feature is not enabled, you do not need to use the inventory detail subrecord to access bin and numbered inventory fields, and you do not need to update any related web services code from prior to 2011.2.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Creating a Work Order by Referencing a Sales Order Line

If appropriate, you can create a work order by referencing a line on a sales order. To use this approach, you supply values for three fields on the work order record:

- **sourceTransactionId** — Use this field to reference the internal ID of the sales order.
- **sourceTransactionLine** — Use this field to reference the line of the sales order for which you want to create the work order.
- **specialOrder** — Set this boolean field to true to indicate that you want the previous two fields to be referenced as the work order is created.

This technique results in the creation of the following links between the sales order and the work order:

- On the line item of the sales order, a link to the work order is created in the **Create WO** column.
- On the work order, a link to the sales order is created in the **Created From** body field.

This approach to creating a work order is similar to the process of creating a work order through initialization. However, note that some body fields on the work order are not automatically populated. For example, you must manually populate the fields for assembly, subsidiary, and quantity.

The following Java example shows how to create a work order using this method:

```
public void testCreateWO() throws Exception
{
   WorkOrder  wo = new WorkOrder();

   wo.setSourceTransactionId("27"); // SalesOrder internalId
   wo.setSourceTransactionLine(1L); // Line Number
   wo.setSpecialOrder(true);  // To create a work order by referencing a sales order, this must
 be true.

   wo.setAssemblyItem(mrr("10"));
   wo.setSubsidiary(mrr("1"));
   wo.setQuantity(1.0);

   String woId = c.addRecord(wo);

   c.getRecord(woId, RecordType.workOrder);
}
```

Work Order Close

If the Manufacturing Work In Process (WIP) feature has been enabled, you can use SuiteTalk to interact with work order close records. You can check to see if WIP is enabled by going to Setup > Company > Enable Features, and reviewing the Items & Inventory tab.

With WIP, instead of creating a single assembly build record to denote that a work order has been addressed, you track progress of the work using three records: work order issue, work order completion, and work order close. This approach lets you manage the assembly process in a more granular way, and to keep the General Ledger up to date as materials move through the different phases of assembly. For more on the benefits of WIP, see the help topic Manufacturing Work In Process (WIP).

The work order close record is used to indicate that all accounting variances that might have arisen during the assembly process have been resolved. In the UI, you can view the form used for creating this record by viewing the work order and clicking the Close button.

The work order issue record is defined in the tranInvt (inventory) XSD.

Supported Operations

The following operations can be used with work order close records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's work order close reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Refer to the following sections for more details on interacting with work order close records.

Prerequisites for Creating a Record

Before you can create a work order close record, a work order record must already exist, and the work order must be configured to use WIP (the WIP box on the work order record must be selected). This is true regardless of whether you are creating the work order close record using initialize and add, or add by itself. If you try to create a work order close record referencing a work order that has *not* been configured to use WIP, the system generates an error reading in part, “One of the following problems exists: You have an invalid work order < *work order ID* >, the work order does not use WIP, or the work order is already closed.”

You can create and modify work orders by choosing Transactions > Manufacturing > Enter Work Orders. You can also interact with work orders using web services, as described in Work Order.

It does not matter if a work order issue record or a work order completion exists for your work order. You can go straight from entering the work order to creating the work order close record.

Using Initialize Versus Add

You can initialize a work order close record from a Work Order record. This approach is the recommended one, though you can also create the record using the add operation by itself.

The initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

If you are using the add operation by itself, note that you must use the createdFrom field to identify the appropriate work order. If you fail to set a value for this field, the system generates an error reading, “Transaction can only be created from a work order.”

The ComponentList Sublist Is Not Returned During a Get

Note that the componentList sublist is not exposed, so performing a get operation on the work order close record will not include the items. This is true even though you can view the sublist in the UI when you display the work order close record there. The sublist is read only, and read-only sublists generally are not exposed.

Sample Code

The following code illustrates how to add a work order close record using a few different techniques.

Using Initialize

This example shows how to create a work order close record using both the initialize and add operations, which is the recommended approach. Note that this technique results in two SOAP requests and two SOAP responses.

Java

```
InitializeRef initRef = new InitializeRef();
initRef.setType(InitializeRefType.workOrder);
initRef.setInternalId("167");

WorkOrderClose woRecord = (WorkOrderClose) c.initialize(initRef,InitializeType.workOrderClose,n
ull);

c.addRecord(woRecord);
```

SOAP Request (Initialize)

```
<soapenv:Body>
   <initialize xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <initializeRecord>
         <ns7:type xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">workOrderClose
</ns7:type>
         <ns8:reference internalId="167" type="workOrder" xmlns:ns8="urn:core_2017_1.platform.w
ebservices.netsuite.com"/>
      </initializeRecord>
   </initialize>
</soapenv:Body>
```

SOAP Response (Initialize)

```
<soapenv:Body>
   <initializeResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <readResponse>
         <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.web
services.netsuite.com"/>
         <record xsi:type="tranInvt:WorkOrderClose" xmlns:tranInvt="urn:inventory_2017_1.transa
ctions.webservices.netsuite.com">
            <tranInvt:createdDate>2013-03-07T07:10:00.000-08:00</tranInvt:createdDate>
            <tranInvt:lastModifiedDate>2013-03-07T08:25:00.000-08:00</tranInvt:lastModifiedDate
>
            <tranInvt:tranId>1</tranInvt:tranId>
            <tranInvt:item internalId="247" xmlns:platformCore="urn:core_2017_1.platform.webser
vices.netsuite.com">
               <platformCore:name>JS Assembly Item Y</platformCore:name>
            </tranInvt:item>
            <tranInvt:orderQuantity>5.0</tranInvt:orderQuantity>
            <tranInvt:createdFrom internalId="167" xmlns:platformCore="urn:core_2017_1.platform
```

```
.webservices.netsuite.com">
               <platformCore:name>Work Order #1</platformCore:name>
            </tranInvt:createdFrom>
            <tranInvt:tranDate>2013-03-07T00:00:00.000-08:00</tranInvt:tranDate>
            <tranInvt:postingPeriod internalId="141" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com">
               <platformCore:name>Mar 2013</platformCore:name>
            </tranInvt:postingPeriod>
            <tranInvt:subsidiary internalId="3" xmlns:platformCore="urn:core_2017_1.platform.we
bservices.netsuite.com">
               <platformCore:name>SUB  UK</platformCore:name>
            </tranInvt:subsidiary>
            <tranInvt:location internalId="2" xmlns:platformCore="urn:core_2017_1.platform.webs
ervices.netsuite.com">
               <platformCore:name>Location UK</platformCore:name>
            </tranInvt:location>
            <tranInvt:customFieldList xmlns:platformCore="urn:core_2017_1.platform.webservices.
netsuite.com">
               <platformCore:customField internalId="43" scriptId="custbody_633637_bsubmit" xsi
:type="platformCore:BooleanCustomFieldRef">
                  <platformCore:value>false</platformCore:value>
               </platformCore:customField>
               <platformCore:customField internalId="165" scriptId="custbody_633637_bload" xsi:
type="platformCore:BooleanCustomFieldRef">
                  <platformCore:value>false</platformCore:value>
               </platformCore:customField>
               <platformCore:customField internalId="65" scriptId="custbody_633637_asubmit" xsi
:type="platformCore:BooleanCustomFieldRef">
                  <platformCore:value>false</platformCore:value>
               </platformCore:customField>
            </tranInvt:customFieldList>
         </record>
      </readResponse>
   </initializeResponse>
</soapenv:Body>
```

SOAP Request (Add)

```
<soapenv:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record xsi:type="ns7:WorkOrderClose" xmlns:ns7="urn:inventory_2017_1.transactions.webser
vices.netsuite.com">
         <ns7:createdDate xsi:type="xsd:dateTime">2013-03-07T15:10:00.000Z</ns7:createdDate>

         <ns7:lastModifiedDate xsi:type="xsd:dateTime">2013-03-07T16:25:00.000Z</ns7:lastModifi
edDate>
         <ns7:tranId xsi:type="xsd:string">1</ns7:tranId>
         <ns7:item internalId="247" xsi:type="ns8:RecordRef" xmlns:ns8="urn:core_2017_1.platfor
m.webservices.netsuite.com">
            <ns8:name xsi:type="xsd:string">JS Assembly Item Y</ns8:name>
         </ns7:item>
         <ns7:orderQuantity xsi:type="xsd:double">5.0</ns7:orderQuantity>
         <ns7:createdFrom internalId="167" xsi:type="ns9:RecordRef" xmlns:ns9="urn:core_2017_1.
platform.webservices.netsuite.com">
            <ns9:name xsi:type="xsd:string">Work Order #1</ns9:name>
         </ns7:createdFrom>
         <ns7:tranDate xsi:type="xsd:dateTime">2013-03-07T08:00:00.000Z</ns7:tranDate>
         <ns7:postingPeriod internalId="141" xsi:type="ns10:RecordRef" xmlns:ns10="urn:core_201
7_1.platform.webservices.netsuite.com">
            <ns10:name xsi:type="xsd:string">Mar 2013</ns10:name>
         </ns7:postingPeriod>
         <ns7:subsidiary internalId="3" xsi:type="ns11:RecordRef" xmlns:ns11="urn:core_2017_1.p
latform.webservices.netsuite.com">
            <ns11:name xsi:type="xsd:string">SUB  UK</ns11:name>
         </ns7:subsidiary>
         <ns7:location internalId="2" xsi:type="ns12:RecordRef" xmlns:ns12="urn:core_2017_1.pla
tform.webservices.netsuite.com">
            <ns12:name xsi:type="xsd:string">Location UK</ns12:name>
         </ns7:location>
         <ns7:customFieldList xsi:type="ns13:CustomFieldList" xmlns:ns13="urn:core_2017_1.platf
orm.webservices.netsuite.com">
            <ns13:customField scriptId="custbody_633637_bsubmit" xsi:type="ns13:BooleanCustomFi
eldRef">
               <ns13:value xsi:type="xsd:boolean">false</ns13:value>
            </ns13:customField>
            <ns13:customField scriptId="custbody_633637_bload" xsi:type="ns13:BooleanCustomFiel
dRef">
               <ns13:value xsi:type="xsd:boolean">false</ns13:value>
            </ns13:customField>
            <ns13:customField scriptId="custbody_633637_asubmit" xsi:type="ns13:BooleanCustomFi
eldRef">
               <ns13:value xsi:type="xsd:boolean">false</ns13:value>
            </ns13:customField>
         </ns7:customFieldList>
      </record>
   </add>
</soapenv:Body>
```

SOAP Response (Add)

```
<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 internalId="270" type="workOrderClose" xsi:type="platformCore:RecordRef" xmln
s:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Using Add

This example shows how to create a work order close record using the add operation alone.

Java

```
RecordRef createdFromRef = new RecordRef();
```

```
createdFromRef.setInternalId("167");

RecordRef postingPeriodRef = new RecordRef();
postingPeriodRef.setInternalId("141");

RecordRef departmentRef = new RecordRef();
departmentRef.setInternalId("2");

RecordRef classRef = new RecordRef();
classRef.setInternalId("2");

WorkOrderClose newWOCl = new WorkOrderClose();
newWOCl.setExternalId("WOCl-JS-001-TEST");
newWOCl.setTranId("TEST WOCl #1");
newWOCl.setCreatedFrom(createdFromRef);
newWOCl.setPostingPeriod(postingPeriodRef);
newWOCl.setDepartment(departmentRef);
newWOCl.set_class(classRef);
newWOCl.setMemo("Memo text");

c.addRecord(newWOCl);
```

SOAP Request

```
<soapenv:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record externalId="WOCl-JS-001-TEST" xsi:type="ns6:WorkOrderClose" xmlns:ns6="urn:invent
ory_2017_1.transactions.webservices.netsuite.com">
         <ns6:tranId xsi:type="xsd:string">TEST WOCl #1</ns6:tranId>
         <ns6:createdFrom internalId="167" xsi:type="ns7:RecordRef" xmlns:ns7="urn:core_2017_1.
platform.webservices.netsuite.com"/>
         <ns6:postingPeriod internalId="141" xsi:type="ns8:RecordRef" xmlns:ns8="urn:core_2017_
1.platform.webservices.netsuite.com"/>
         <ns6:memo xsi:type="xsd:string">Memo text</ns6:memo>
         <ns6:department internalId="2" xsi:type="ns9:RecordRef" xmlns:ns9="urn:core_2017_1.pla
tform.webservices.netsuite.com"/>
         <ns6:class internalId="2" xsi:type="ns10:RecordRef" xmlns:ns10="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
      </record>
   </add>
</soapenv: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 internalId="269" externalId="WOCl-JS-001-TEST" type="workOrderClose" xsi:type
="platformCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com
"/>
      </writeResponse>
   </addResponse>
```

```
</soapenv:Body>
```

Work Order Completion

If the Manufacturing Work In Process (WIP) feature has been enabled, you can use SuiteTalk to interact with work order completion records. You can check to see if WIP is enabled by going to Setup > Company > Enable Features, and reviewing the Items & Inventory tab.

With WIP, instead of creating a single assembly build record to denote that a work order has been addressed, you track progress of the work using three records: work order issue, work order completion, and work order close. This approach lets you manage the assembly process in a more granular way, and to keep the General Ledger up to date as materials move through the different phases of assembly. For more on the benefits of WIP, refer to Manufacturing Work In Process (WIP).

The work order completion record is used to indicate that assemblies have been built. You can also optionally use this record to record that raw materials — items in the componentList sublist — have been consumed as part of the assembly process. This latter option is called entering a completion with backflush. For example, you might enter a completion with backflush if previous records (such as work order issue) did not record the consumption of all the materials you ended up using.

In the UI, you can view the form used for creating the work order completion record by choosing Transactions > Manufacturing > Enter Completions, selecting a Subsidiary (for OneWorld accounts), then clicking the Complete link that corresponds with one of the listed work orders. An alternate method is to view the work order and click one of two buttons: Enter Completions or Enter Completions With Backflush. For help filling out the form manually, refer to Entering Work Order Completions.

The work order completion record is defined in the tranInvt (inventory) XSD.

Supported Operations

The following operations can be used with work order completion records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | initialize / initializeList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's work order completion reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Refer to the following sections for more details on interacting with work order completion records.

Prerequisites for Creating a Record

Before you can create a work order completion record, a work order record must already exist, and the work order must be configured to use WIP (the WIP box on the work order record must be selected). This is true regardless of whether you are creating the work order issue record using initialize and add, or add by itself. If you try to create a work order issue record referencing a work order that has *not* been configured to use WIP, the system generates an error reading in part, “One of the following problems exists: You have an invalid work order < *work order ID* >, the work order does not use WIP, or the work order is already closed.”

You can create work orders by choosing Transactions > Manufacturing > Enter Work Orders. You can also interact with work orders using web services, as described in Work Order.

Further, the assembly item referenced in the work order must be properly set up for WIP. For details on this process, see the help topic Setting Up Items as WIP Assemblies.

Note that it does not matter if a work order issue record already exists for your work order. You can go straight from entering the work order to creating the work order completion record.

Using Initialize Versus Add

You can initialize a work order completion record from a Work Order record. This approach is the recommended one, though you can also create the record using the add operation by itself.

Using Both Initialize and Add

The initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

Note that when you use the initialize operation, certain required fields are not initialized and must be set manually. These fields vary depending on the work order that you are using, as follows:

- If the work order uses routing, the startOperation and endOperation fields are both required (but are not initialized). These fields denote which operation tasks were completed.
- If the work order does not use routing, the quantity field is required (but is not initialized). The quantity element denotes the total number of assembly items that were completed in the build process. Note that you cannot update this field to a value greater than the value called for in the work order. Attempting to set the value too high results in an error.

In each case, you must manually set the required values in a statement between the initialize and add statements. To see an example, refer to Using Initialize and Setting isBackflush to False.

Using Add

If you are using the add operation by itself, note that you must use the createdFrom field to identify the appropriate work order. If you fail to set a value for this field, the system generates an error reading, “Transaction can only be created from a work order.”

Working with the OperationList Sublist

When routing is used, an additional sublist, operationList, is available. If the original work order does not use routing, the operationList sublist is unavailable.

Working with the ComponentList Sublist

When working with the componentList sublist, note that the same general restrictions apply as when you are using the work order issue record. For details, see Working with the ComponentList Sublist.

In addition, note that, with work order completion, you can interact with the componentList sublist *only* if the isBackflush element is set to true.

Refer also to Using ReplaceAll for details on the validation that occurs when replaceAll is set to true.

Sample Code

The following code illustrates how to create a work order completion record using a few different techniques.

Using Initialize and Setting isBackflush to False

The recommended strategy of creating a work order completion record is to use the initialize operation. Note that this approach generates two sets of SOAP requests and responses — one for the initialize operation and one for the add operation. This example shows how to use initialize to create a work order completion with isBackflush set to false.

Note that isBackflush is set to false by default. To include quantity values for items on the componentList, you would have to set isBackflush to true and specify the quantity values for the component items. (To see an example that sets isBackflush to true, see Using Add and Setting isBackflush to True.)

Further, note that in this example, the work order being referenced does not use routing. For this reason, the quantity value is required — and because the value is not initialized, it must be manually set. You should set the quantity value between the initialize and add statements.

Java

```
InitializeRef initRef = new InitializeRef();
initRef.setType(InitializeRefType.workOrder);
initRef.setInternalId("167");

WorkOrderCompletion woRecord = (WorkOrderCompletion) c.initialize(initRef,InitializeType.workOr
derCompletion,null);

woRecord.setQuantity(3.0);

c.addRecord(woRecord);
```

SOAP Request (Initialize)

```
<soapenv:Body>
   <initialize xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <initializeRecord>
         <ns7:type xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">workOrderCompl
etion</ns7:type>
         <ns8:reference internalId="167" type="workOrder" xmlns:ns8="urn:core_2017_1.platform.w
ebservices.netsuite.com"/>
```

```
      </initializeRecord>
   </initialize>
</soapenv:Body>
```

SOAP Response (Initialize)

```
<soapenv:Body>
   <initializeResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <readResponse>
         <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.web
services.netsuite.com"/>
         <record xsi:type="tranInvt:WorkOrderCompletion" xmlns:tranInvt="urn:inventory_2017_1.t
ransactions.webservices.netsuite.com">
            <tranInvt:createdDate>2013-03-07T07:10:00.000-08:00</tranInvt:createdDate>
            <tranInvt:lastModifiedDate>2013-03-07T08:14:00.000-08:00</tranInvt:lastModifiedDate
>
            <tranInvt:tranId>2</tranInvt:tranId>
            <tranInvt:item internalId="247" xmlns:platformCore="urn:core_2017_1.platform.webser
vices.netsuite.com">
               <platformCore:name>JS Assembly Item Y</platformCore:name>
            </tranInvt:item>
            <tranInvt:isBackflush>false</tranInvt:isBackflush>
            <tranInvt:orderQuantity>5.0</tranInvt:orderQuantity>
            <tranInvt:createdFrom internalId="167" xmlns:platformCore="urn:core_2017_1.platform
.webservices.netsuite.com">
               <platformCore:name>Work Order #1</platformCore:name>
            </tranInvt:createdFrom>
            <tranInvt:tranDate>2013-03-07T00:00:00.000-08:00</tranInvt:tranDate>
            <tranInvt:postingPeriod internalId="141" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com">
               <platformCore:name>Mar 2013</platformCore:name>
            </tranInvt:postingPeriod>
            <tranInvt:subsidiary internalId="3" xmlns:platformCore="urn:core_2017_1.platform.we
bservices.netsuite.com">
               <platformCore:name>SUB  UK</platformCore:name>
            </tranInvt:subsidiary>
            <tranInvt:location internalId="2" xmlns:platformCore="urn:core_2017_1.platform.webs
ervices.netsuite.com">
               <platformCore:name>Location UK</platformCore:name>
      </tranInvt:location>
            <tranInvt:componentList>
               <tranInvt:workOrderCompletionComponent>
                  <tranInvt:item internalId="245" xmlns:platformCore="urn:core_2017_1.platform.
webservices.netsuite.com"/>
                  <tranInvt:quantityPer>5.0</tranInvt:quantityPer>
                  <tranInvt:lineNumber>5</tranInvt:lineNumber>
               </tranInvt:workOrderCompletionComponent>
                  <tranInvt:workOrderCompletionComponent>
                  <tranInvt:item internalId="246" xmlns:platformCore="urn:core_2017_1.platform.
webservices.netsuite.com"/>
                  <tranInvt:quantityPer>6.0</tranInvt:quantityPer>
                  <tranInvt:lineNumber>6</tranInvt:lineNumber>
               </tranInvt:workOrderCompletionComponent>
            </tranInvt:componentList>
            <tranInvt:customFieldList xmlns:platformCore="urn:core_2017_1.platform.webservices.
```

```
netsuite.com">
               <platformCore:customField internalId="65" scriptId="custbody_633637_bsubmit" xsi
:type="platformCore:BooleanCustomFieldRef">
                  <platformCore:value>false</platformCore:value>
               </platformCore:customField>
               <platformCore:customField internalId="215" scriptId="custbody_633637_bload" xsi:
type="platformCore:BooleanCustomFieldRef">
                  <platformCore:value>false</platformCore:value>
               </platformCore:customField>
               <platformCore:customField internalId="53" scriptId="custbody_633637_asubmit" xsi
:type="platformCore:BooleanCustomFieldRef">
                  <platformCore:value>false</platformCore:value>
               </platformCore:customField>
            </tranInvt:customFieldList>
         </record>
      </readResponse>
   </initializeResponse>
</soapenv:Body>
```

SOAP Request (Add)

```
<soapenv:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record xsi:type="ns7:WorkOrderCompletion" xmlns:ns7="urn:inventory_2017_1.transactions.w
ebservices.netsuite.com">
         <ns7:createdDate xsi:type="xsd:dateTime">2013-03-07T15:10:00.000Z</ns7:createdDate>

         <ns7:lastModifiedDate xsi:type="xsd:dateTime">2013-03-07T16:14:00.000Z</ns7:lastModifi
edDate>
         <ns7:tranId xsi:type="xsd:string">2</ns7:tranId>
         <ns7:item internalId="247" xsi:type="ns8:RecordRef" xmlns:ns8="urn:core_2017_1.platfor
m.webservices.netsuite.com">
            <ns8:name xsi:type="xsd:string">JS Assembly Item Y</ns8:name>
         </ns7:item>
         <ns7:quantity xsi:type="xsd:double">3.0</ns7:quantity>
         <ns7:isBackflush xsi:type="xsd:boolean">false</ns7:isBackflush>
         <ns7:orderQuantity xsi:type="xsd:double">5.0</ns7:orderQuantity>
         <ns7:createdFrom internalId="167" xsi:type="ns9:RecordRef" xmlns:ns9="urn:core_2017_1.
platform.webservices.netsuite.com">
            <ns9:name xsi:type="xsd:string">Work Order #1</ns9:name>
         </ns7:createdFrom>
         <ns7:tranDate xsi:type="xsd:dateTime">2013-03-07T08:00:00.000Z</ns7:tranDate>
         <ns7:postingPeriod internalId="141" xsi:type="ns10:RecordRef" xmlns:ns10="urn:core_201
7_1.platform.webservices.netsuite.com">
            <ns10:name xsi:type="xsd:string">Mar 2013</ns10:name>
         </ns7:postingPeriod>
         <ns7:subsidiary internalId="3" xsi:type="ns11:RecordRef" xmlns:ns11="urn:core_2017_1.p
latform.webservices.netsuite.com">
            <ns11:name xsi:type="xsd:string">SUB  UK</ns11:name>
         </ns7:subsidiary>
         <ns7:location internalId="2" xsi:type="ns12:RecordRef" xmlns:ns12="urn:core_2017_1.pla
tform.webservices.netsuite.com">
            <ns12:name xsi:type="xsd:string">Location UK</ns12:name>
         </ns7:location>
         <ns7:componentList replaceAll="false" xsi:type="ns7:WorkOrderCompletionComponentList">
```

```
        <ns7:workOrderCompletionComponent xsi:type="ns7:WorkOrderCompletionComponent">
        <ns7:item internalId="245" xsi:type="ns13:RecordRef" xmlns:ns13="urn:core_2017_1.platf
orm.webservices.netsuite.com"/>
        <ns7:quantityPer xsi:type="xsd:double">5.0</ns7:quantityPer>
        <ns7:lineNumber xsi:type="xsd:long">5</ns7:lineNumber>
        </ns7:workOrderCompletionComponent>
        <ns7:workOrderCompletionComponent xsi:type="ns7:WorkOrderCompletionComponent">
        <ns7:item internalId="246" xsi:type="ns14:RecordRef" xmlns:ns14="urn:core_2017_1.platf
orm.webservices.netsuite.com"/>
        <ns7:quantityPer xsi:type="xsd:double">6.0</ns7:quantityPer>
        <ns7:lineNumber xsi:type="xsd:long">6</ns7:lineNumber>
        </ns7:workOrderCompletionComponent>
        </ns7:componentList>
        <ns7:customFieldList xsi:type="ns15:CustomFieldList" xmlns:ns15="urn:core_2017_1.platf
orm.webservices.netsuite.com">
          <ns15:customField internalId="65" scriptId="custbody_633637_bsubmit" xsi:type="ns15
:BooleanCustomFieldRef">
            <ns15:value xsi:type="xsd:boolean">false</ns15:value>
          </ns15:customField>
          <ns15:customField internalId="215" scriptId="custbody_633637_bload" xsi:type="ns15:
BooleanCustomFieldRef">
            <ns15:value xsi:type="xsd:boolean">false</ns15:value>
          </ns15:customField>
          <ns15:customField internalId="53" scriptId="custbody_633637_asubmit" xsi:type="ns15
:BooleanCustomFieldRef">
            <ns15:value xsi:type="xsd:boolean">false</ns15:value>
          </ns15:customField>
        </ns7:customFieldList>
      </record>
   </add>
</soapenv:Body>
```

SOAP Response (Add)

```
<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 internalId="268" type="workOrderCompletion" xsi:type="platformCore:RecordRef"
 xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Using Add and Setting isBackflush to True

The following example omits the initialize operation — using only the add operation to create the record. In this example, isBackflush is set to true and values are sent for the quantities of items in the componentList sublist.

Note that this example sets a value for the quantity body field. However, if the original work order used a routing, the quantity field would not be updatable.

Java

```
RecordRef createdFromRef = new RecordRef();
createdFromRef.setInternalId("167");

RecordRef postingPeriodRef = new RecordRef();
postingPeriodRef.setInternalId("141");

RecordRef departmentRef = new RecordRef();
departmentRef.setInternalId("2");

RecordRef classRef = new RecordRef();
classRef.setInternalId("2");

RecordRef item1Ref = new RecordRef();
item1Ref.setInternalId("245");

RecordRef item2Ref = new RecordRef();
item2Ref.setInternalId("246");

WorkOrderCompletion newWOCo = new WorkOrderCompletion();
newWOCo.setExternalId("WOCo-JS-001-TEST-BFT");
newWOCo.setCreatedFrom(createdFromRef);
newWOCo.setQuantity(2.0);
newWOCo.setIsBackflush(true);
newWOCo.setPostingPeriod(postingPeriodRef);
newWOCo.setDepartment(departmentRef);
newWOCo.set_class(classRef);
newWOCo.setMemo("Add - isBackFlush = T");

WorkOrderCompletionComponent[] componentListArray = { new WorkOrderCompletionComponent(), new W
orkOrderCompletionComponent() };
componentListArray[0].setItem(item1Ref);
componentListArray[0].setQuantity(3.0);
componentListArray[1].setItem(item2Ref);
componentListArray[1].setQuantity(4.0);

WorkOrderCompletionComponentList componentList = new WorkOrderCompletionComponentList();
componentList.setWorkOrderCompletionComponent(componentListArray);
newWOCo.setComponentList(componentList);

c.addRecord(newWOCo);
```

SOAP Request

```
<soapenv:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record externalId="WOCo-JS-001-TEST-BFT" xsi:type="ns6:WorkOrderCompletion" xmlns:ns6="u
rn:inventory_2017_1.transactions.webservices.netsuite.com">
         <ns6:quantity xsi:type="xsd:double">2.0</ns6:quantity>
         <ns6:isBackflush xsi:type="xsd:boolean">true</ns6:isBackflush>
         <ns6:createdFrom internalId="167" xsi:type="ns7:RecordRef" xmlns:ns7="urn:core_2017_1.
platform.webservices.netsuite.com"/>
         <ns6:postingPeriod internalId="141" xsi:type="ns8:RecordRef" xmlns:ns8="urn:core_2017_
```

```
1.platform.webservices.netsuite.com"/>
         <ns6:memo xsi:type="xsd:string">Add - isBackFlush = T</ns6:memo>
         <ns6:department internalId="2" xsi:type="ns9:RecordRef" xmlns:ns9="urn:core_2017_1.pla
tform.webservices.netsuite.com"/>
         <ns6:class internalId="2" xsi:type="ns10:RecordRef" xmlns:ns10="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
         <ns6:componentList replaceAll="false" xsi:type="ns6:WorkOrderCompletionComponentList">

            <ns6:workOrderCompletionComponent xsi:type="ns6:WorkOrderCompletionComponent">

               <ns6:item internalId="245" xsi:type="ns11:RecordRef" xmlns:ns11="urn:core_2017_1
.platform.webservices.netsuite.com"/>
               <ns6:quantity xsi:type="xsd:double">3.0</ns6:quantity>
            </ns6:workOrderCompletionComponent>
            <ns6:workOrderCompletionComponent xsi:type="ns6:WorkOrderCompletionComponent">

               <ns6:item internalId="246" xsi:type="ns12:RecordRef" xmlns:ns12="urn:core_2017_1
.platform.webservices.netsuite.com"/>
               <ns6:quantity xsi:type="xsd:double">4.0</ns6:quantity>
            </ns6:workOrderCompletionComponent>
         </ns6:componentList>
      </record>
   </add>
</soapenv: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 internalId="267" externalId="WOCo-JS-001-TEST-BFT" type="workOrderCompletion"
 xsi:type="platformCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.net
suite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Work Order Issue

If the Manufacturing Work In Process (WIP) feature has been enabled, you can use SuiteTalk to interact with work order issue records. You can check to see if WIP is enabled by going to Setup > Company > Enable Features, and reviewing the Items & Inventory tab.

With WIP, instead of creating a single assembly build record to denote that a work order has been addressed, you track progress of the work using three records: work order issue, work order completion, and work order close. This approach lets you manage the assembly process in a more granular way, and to keep the General Ledger up to date as materials move through the different phases of assembly. For more on the benefits of WIP, refer to Manufacturing Work In Process (WIP).

The work order issue record is used to indicate that particular quantities of raw materials, or component items, have been gathered for the production of the assembly item (or items).

In the UI, you can view the form used for creating the work order issue record by choosing Transactions > Manufacturing > Issue Components, selecting a Subsidiary (for OneWorld accounts), then clicking the Issue link that corresponds with one of listed work orders. An alternate method is to view the work order and click the Issue Components button. For help filling out the form manually, refer to Entering Work Order Issues.

The work order issue record is defined in the tranInvt (inventory) XSD.

Supported Operations

The following operations can be used with work order issue records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | initialize / initializeList | search | searchMore | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's work order issue reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Refer to the following sections for more details on interacting with work order issue records.

Prerequisites for Creating a Record

Before you can create a work order issue record, a work order record must already exist, and the work order must be configured to use WIP (the WIP box on the work order record must be selected). This is true regardless of whether you are creating the work order issue record using initialize and add, or add by itself. If you try to create a work order issue record referencing a work order that has *not* been configured to use WIP, the system generates an error reading in part, "One of the following problems exists: You have an invalid work order < *work order ID* >, the work order does not use WIP, or the work order is already closed."

You can create and modify work orders by choosing Transactions > Manufacturing > Enter Work Orders. You can also interact with work orders using web services, as described in Work Order.

Note also that the assembly item referenced in the work order must be properly set up for WIP. as described in the Setting Up Items as WIP Assemblies.

Using Initialize Versus Add

You can initialize a work order issue record from a Work Order record. This approach is the recommended one, though you can also create the record using the add operation by itself.

The initialize operation emulates the UI workflow by prepopulating fields on transaction line items with values from a related record. For more information about this operation, see the help topic initialize / initializeList.

If you are using the add operation by itself, note that you must use the createdFrom field to identify the appropriate work order. If you fail to set a value for this field, the system generates an error reading, “Transaction can only be created from a work order.”

Working with the ComponentList Sublist

The work order issue record includes the componentList sublist, which identifies the component items required by the work order.

When working with the work order issue record, most of the data in the componentList sublist is static — except for quantity. That is, you can update the individual quantities of each of the items listed, but you cannot add or remove items from the list.

In regard to quantity adjustments, note that, for each item, there is a maximum quantity. This maximum is determined by both the assembly item record and the work order record. For example, suppose you have an assembly item record for a widget that calls for 10 nails. If you create a work order calling for six builds of the widget, a total of 60 nails will be needed. Therefore, when you create the work order issue record, you can update the quantity of nails up to a maximum of 60.

Related to this, note that the maximum is cumulative. This rule is significant if you create multiple work order issue records for the same work order, or if you have also created work order completions with backflush (another record that lets you specify that certain quantities of component items have been used). To follow on with the previous example, suppose you create one work order issue record that records the use of 10 nails, and one work order completion with backflush that records the use of 20 nails. An additional work order issue record could denote the use of only 30 nails.

Using ReplaceAll

When you set replaceAll to true when working with the componentList sublist, you must specify a value for each record in the sublist. If you fail to reference each record, the system generates an error.

Sample Code

The following code illustrates how to add a work order issue record using a few different techniques.

Using Both Initialize and Add

This example shows how to create a work order issue record using both the initialize and add operations, which is the recommended approach. Note that this technique results in two sets of SOAP requests and responses.

Although the sample below does not change the quantities of items on the componentList sublist, it would be possible to do so by using set commands between the initialize and add statements.

Java

```
InitializeRef initRef = new InitializeRef();
initRef.setType(InitializeRefType.workOrder);
initRef.setInternalId("167");

WorkOrderIssue woRecord = (WorkOrderIssue) c.initialize(initRef,InitializeType.workOrderIssue,n
ull);
```

```
c.addRecord(woRecord);
```

SOAP Request (Initialize)

```
<soapenv:Body>
    <initialize xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <initializeRecord>
         <ns7:type xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">workOrderIssue
</ns7:type>
         <ns8:reference internalId="167" type="workOrder" xmlns:ns8="urn:core_2017_1.platform.w
ebservices.netsuite.com"/>
      </initializeRecord>
   </initialize>
</soapenv:Body>
```

SOAP Response (Initialize)

```
<soapenv:Body>
    <initializeResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
       <readResponse>
         <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.web
services.netsuite.com"/>
         <record xsi:type="tranInvt:WorkOrderIssue" xmlns:tranInvt="urn:inventory_2017_1.transa
ctions.webservices.netsuite.com">
            <tranInvt:createdDate>2013-03-07T07:10:00.000-08:00</tranInvt:createdDate>
            <tranInvt:lastModifiedDate>2013-03-07T07:43:00.000-08:00</tranInvt:lastModifiedDate
>
            <tranInvt:tranId>3</tranInvt:tranId>
              <tranInvt:item internalId="247" xmlns:platformCore="urn:core_2017_1.platform.webs
ervices.netsuite.com">
               <platformCore:name>JS Assembly Item Y</platformCore:name>
            </tranInvt:item>
            <tranInvt:createdFrom internalId="167" xmlns:platformCore="urn:core_2017_1.platform
.webservices.netsuite.com">
               <platformCore:name>Work Order #1</platformCore:name>
            </tranInvt:createdFrom>
            <tranInvt:tranDate>2013-03-07T00:00:00.000-08:00</tranInvt:tranDate>
            <tranInvt:postingPeriod internalId="141" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com">
               <platformCore:name>Mar 2013</platformCore:name>
            </tranInvt:postingPeriod>
            <tranInvt:subsidiary internalId="3" xmlns:platformCore="urn:core_2017_1.platform.we
bservices.netsuite.com">
               <platformCore:name>SUB  UK</platformCore:name>
            </tranInvt:subsidiary>
            <tranInvt:location internalId="2" xmlns:platformCore="urn:core_2017_1.platform.webs
ervices.netsuite.com">
               <platformCore:name>Location UK</platformCore:name>
            </tranInvt:location>
            <tranInvt:componentList>
               <tranInvt:workOrderIssueComponent>
                  <tranInvt:item internalId="245" xmlns:platformCore="urn:core_2017_1.platform.
webservices.netsuite.com"/>
```

```
                   <tranInvt:quantity>23.0</tranInvt:quantity>
                    <tranInvt:lineNumber>5</tranInvt:lineNumber>
                </tranInvt:workOrderIssueComponent>
                   <tranInvt:workOrderIssueComponent>
                   <tranInvt:item internalId="246" xmlns:platformCore="urn:core_2017_1.platform.
webservices.netsuite.com"/>
                   <tranInvt:quantity>27.0</tranInvt:quantity>
                   <tranInvt:lineNumber>6</tranInvt:lineNumber>
                </tranInvt:workOrderIssueComponent>
             </tranInvt:componentList>
             <tranInvt:customFieldList xmlns:platformCore="urn:core_2017_1.platform.webservices.
netsuite.com">
                <platformCore:customField internalId="65" scriptId="custbody_633637_bsubmit" xsi
:type="platformCore:BooleanCustomFieldRef">
                   <platformCore:value>false</platformCore:value>
                </platformCore:customField>
                <platformCore:customField internalId="34" scriptId="custbody_633637_bload" xsi:t
ype="platformCore:BooleanCustomFieldRef">
                   <platformCore:value>false</platformCore:value>
                </platformCore:customField>
                <platformCore:customField internalId="215" scriptId="custbody_633637_asubmit" xs
i:type="platformCore:BooleanCustomFieldRef">
                   <platformCore:value>false</platformCore:value>
                </platformCore:customField>
             </tranInvt:customFieldList>
          </record>
       </readResponse>
    </initializeResponse>
</soapenv:Body>
```

SOAP Request (Add)

```
<soapenv:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record xsi:type="ns7:WorkOrderIssue" xmlns:ns7="urn:inventory_2017_1.transactions.webser
vices.netsuite.com">
         <ns7:createdDate xsi:type="xsd:dateTime">2013-03-07T15:10:00.000Z</ns7:createdDate>

         <ns7:lastModifiedDate xsi:type="xsd:dateTime">2013-03-07T15:43:00.000Z</ns7:lastModifi
edDate>
         <ns7:tranId xsi:type="xsd:string">3</ns7:tranId>
         <ns7:item internalId="247" xsi:type="ns8:RecordRef" xmlns:ns8="urn:core_2017_1.platfor
m.webservices.netsuite.com">
            <ns8:name xsi:type="xsd:string">JS Assembly Item Y</ns8:name>
         </ns7:item>
         <ns7:createdFrom internalId="167" xsi:type="ns9:RecordRef" xmlns:ns9="urn:core_2017_1.
platform.webservices.netsuite.com">
            <ns9:name xsi:type="xsd:string">Work Order #1</ns9:name>
         </ns7:createdFrom>
         <ns7:tranDate xsi:type="xsd:dateTime">2013-03-07T08:00:00.000Z</ns7:tranDate>
         <ns7:postingPeriod internalId="141" xsi:type="ns10:RecordRef" xmlns:ns10="urn:core_201
7_1.platform.webservices.netsuite.com">
            <ns10:name xsi:type="xsd:string">Mar 2013</ns10:name>
         </ns7:postingPeriod>
         <ns7:subsidiary internalId="3" xsi:type="ns11:RecordRef" xmlns:ns11="urn:core_2017_1.p
```

```
latform.webservices.netsuite.com">
            <ns11:name xsi:type="xsd:string">SUB  UK</ns11:name>
         </ns7:subsidiary>
         <ns7:location internalId="2" xsi:type="ns12:RecordRef" xmlns:ns12="urn:core_2017_1.pla
tform.webservices.netsuite.com">
            <ns12:name xsi:type="xsd:string">Location UK</ns12:name>
         </ns7:location>
         <ns7:componentList replaceAll="false" xsi:type="ns7:WorkOrderIssueComponentList">

         <ns7:workOrderIssueComponent xsi:type="ns7:WorkOrderIssueComponent">
            <ns7:item internalId="245" xsi:type="ns13:RecordRef" xmlns:ns13="urn:core_2017_1.pl
atform.webservices.netsuite.com"/>
            <ns7:quantity xsi:type="xsd:double">23.0</ns7:quantity>
            <ns7:lineNumber xsi:type="xsd:long">5</ns7:lineNumber>
         </ns7:workOrderIssueComponent>
         <ns7:workOrderIssueComponent xsi:type="ns7:WorkOrderIssueComponent">
            <ns7:item internalId="246" xsi:type="ns14:RecordRef" xmlns:ns14="urn:core_2017_1.pl
atform.webservices.netsuite.com"/>
            <ns7:quantity xsi:type="xsd:double">27.0</ns7:quantity>
            <ns7:lineNumber xsi:type="xsd:long">6</ns7:lineNumber>
         </ns7:workOrderIssueComponent>
         </ns7:componentList>
         <ns7:customFieldList xsi:type="ns15:CustomFieldList" xmlns:ns15="urn:core_2017_1.platf
orm.webservices.netsuite.com">
            <ns15:customField internalId="65" scriptId="custbody_633637_bsubmit" xsi:type="ns15
:BooleanCustomFieldRef">
               <ns15:value xsi:type="xsd:boolean">false</ns15:value>
            </ns15:customField>
            <ns15:customField internalId="215" scriptId="custbody_633637_bload" xsi:type="ns15:
BooleanCustomFieldRef">
               <ns15:value xsi:type="xsd:boolean">false</ns15:value>
            </ns15:customField>
            <ns15:customField internalId="23" scriptId="custbody_633637_asubmit" xsi:type="ns15
:BooleanCustomFieldRef">
               <ns15:value xsi:type="xsd:boolean">false</ns15:value>
            </ns15:customField>
         </ns7:customFieldList>
      </record>
   </add>
</soapenv:Body>
```

SOAP Response (Add)

```
<soapenv:Body>
   <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <writeResponse>
      <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.webser
vices.netsuite.com"/>
      <baseRef internalId="174" type="workOrderIssue" xsi:type="platformCore:RecordRef" xmlns:p
latformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Using Add

This example uses the add operation to create the work order issue record. This example also sets values for the quantities of items in the componentList sublist.

Java

```
RecordRef createdFromRef = new RecordRef();
createdFromRef.setInternalId("167");

RecordRef postingPeriodRef = new RecordRef();
postingPeriodRef.setInternalId("141");

RecordRef departmentRef = new RecordRef();
departmentRef.setInternalId("2");

RecordRef classRef = new RecordRef();
classRef.setInternalId("2");

RecordRef locationRef = new RecordRef();
locationRef.setInternalId("2");

RecordRef item1Ref = new RecordRef();
item1Ref.setInternalId("245");

RecordRef item2Ref = new RecordRef();
item2Ref.setInternalId("246");

WorkOrderIssue newWOI = new WorkOrderIssue();
newWOI.setExternalId("WOI-JS-002-TEST");
newWOI.setCreatedFrom(createdFromRef);
newWOI.setPostingPeriod(postingPeriodRef);
newWOI.setDepartment(departmentRef);
newWOI.set_class(classRef);
newWOI.setLocation(locationRef);
newWOI.setMemo("Memo text");

WorkOrderIssueComponent[] componentListArray = { new WorkOrderIssueComponent(), new WorkOrderIs
sueComponent() };
componentListArray[0].setItem(item1Ref);
componentListArray[0].setQuantity(2.0);
componentListArray[1].setItem(item2Ref);
componentListArray[1].setQuantity(3.0);

WorkOrderIssueComponentList componentList = new WorkOrderIssueComponentList();
componentList.setWorkOrderIssueComponent(componentListArray);
componentList.setReplaceAll(false);
newWOI.setComponentList(componentList);

c.addRecord(newWOI);
```

SOAP Request

```
<soapenv:Body>
```

```
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record externalId="WOI-JS-002-TEST" xsi:type="ns6:WorkOrderIssue" xmlns:ns6="urn:invento
ry_2017_1.transactions.webservices.netsuite.com">
         <ns6:createdFrom internalId="167" xsi:type="ns7:RecordRef" xmlns:ns7="urn:core_2017_1.
platform.webservices.netsuite.com"/>
         <ns6:postingPeriod internalId="141" xsi:type="ns8:RecordRef" xmlns:ns8="urn:core_2017_
1.platform.webservices.netsuite.com"/>
         <ns6:memo xsi:type="xsd:string">Memo text</ns6:memo>
         <ns6:department internalId="2" xsi:type="ns9:RecordRef" xmlns:ns9="urn:core_2017_1.pla
tform.webservices.netsuite.com"/>
         <ns6:class internalId="2" xsi:type="ns10:RecordRef" xmlns:ns10="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
         <ns6:location internalId="2" xsi:type="ns11:RecordRef" xmlns:ns11="urn:core_2017_1.pla
tform.webservices.netsuite.com"/>
         <ns6:componentList replaceAll="false" xsi:type="ns6:WorkOrderIssueComponentList">

            <ns6:workOrderIssueComponent xsi:type="ns6:WorkOrderIssueComponent">
               <ns6:item internalId="245" xsi:type="ns12:RecordRef" xmlns:ns12="urn:core_2017_1
.platform.webservices.netsuite.com"/>
               <ns6:quantity xsi:type="xsd:double">2.0</ns6:quantity>
            </ns6:workOrderIssueComponent>
            <ns6:workOrderIssueComponent xsi:type="ns6:WorkOrderIssueComponent">
               <ns6:item internalId="246" xsi:type="ns13:RecordRef" xmlns:ns13="urn:core_2017_1
.platform.webservices.netsuite.com"/>
               <ns6:quantity xsi:type="xsd:double">3.0</ns6:quantity>
            </ns6:workOrderIssueComponent>
         </ns6:componentList>
      </record>
   </add>
</soapenv: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 internalId="171" externalId="WOI-JS-002-TEST" type="workOrderIssue" xsi:type
="platformCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com
"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Multiple Shipping Routes in Web Services

The following topics are covered in this section. If you are unfamiliar with the Multiple Shipping Routes feature, it is recommended that you read these topics in order.

- What is the Multiple Shipping Routes feature?
- How does MSR work in Web Services?

- Which fields and elements are associated with MSR?
- Does MSR work on custom forms?
- Multiple Shipping Routes Code Samples for Web Services

What is the Multiple Shipping Routes feature?

The Multiple Shipping Routes (MSR) feature lets you associate several items with one transaction, and then set different shipping addresses and shipping methods for each item. Transaction types such as sales order, cash sale, invoice, estimate, and item fulfillment all support MSR.

When MSR is enabled in your account, **and** the Enable Item Line Shipping check box is selected on the transaction, each item can have its own shipping address and shipping method. The shipping address is specified in the Ship To column. The shipping method is specified in the Ship Via column.

In the UI, after all items have been added to the transaction, you must then create individual shipping groups by clicking the **Calculate Shipping** button on the Shipping tab. A shipping group includes all item details related to where the item is being shipped from, where it is being shipped to, the item's shipping method, and its shipping rate.

Although there is no UI label called "Shipping Group," web services developers can verify that shipping groups have been created by doing a *get* on the transaction after it has been submitted.

Note: For additional information on MSR, as well as steps for enabling MSR in your account, see Multiple Shipping Routes in the NetSuite Help Center.

How does MSR work in Web Services?

Generally speaking, NetSuite web services attempts to programmatically mirror the actions a user takes in the UI. However, when working with MSR-enabled transactions in web services, developers should be aware of the following UI vs. Web services distinctions:

- If you attempt to set items and shipping rates in the same request, the following message is returned:

  `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.
- If you make any update to any item on MSR-enabled transactions, this action may result in changes to the shipping cost. Every time an item is updated and the record is submitted, NetSuite re-calculates the shipping rate. NetSuite calculates all orders based on "real-time" shipping rates.
- There is no web services equivalent of the Calculate Shipping button that appears on the Shipping subtab. In web services, shipping calculations are handled by the NetSuite backend when the transaction is submitted. Note that you can retrieve the shipping cost if you do a *get* on the transaction with the *bodyFieldsOnly* search preference set to false.
- The only *initialize* workflow that is impacted with MSR enabled is the sales order to fulfillment workflow. Invoicing and other transaction workflows are not impacted.
- The initialize operation includes an optional argument called *AuxReference*. In *AuxReference* users can set the type to *shippingGroup* by using the enum object InitializeAuxRefType. When working with MSR-enabled transactions, you must specify a value for *shippingGroup*. If you do not specify a value, the value **1** (for the first shipping group) is defaulted. This means that only the items belonging to the first shipping group will be fulfilled when the sales order is initialized. For a code sample, see Initialize the sales order to create the item fulfillment.

Which fields and elements are associated with MSR?

The following table lists UI field labels and their corresponding web services elements for all MSR-related fields.

UI Label	Element Name	Note
Enable Item Line Shipping	isMultishipTo	Set to *true* to allow for multiple items with separate shipping address/methods
Ship To	shipAddress	References the internal ID of the customer's shipping address. You can get the internal ID by clicking the Address tab on the customer's record. The address ID appears in the ID column. **Note:** The Show Internal ID preference must be enabled for address IDs to show.
Ship Via	shipMethod	References the internal ID of the item. Go to the Items list to see all item IDs. **Note:** The Show Internal ID preference must be enabled for address IDs to show.
There is no UI label for Shipping Group.	shipGroup	Each item that has a separate shipping address/shipping method will have a unique shipGroup number. When you initialize a sales order to create a fulfillment, and the sales order has MSR enabled, you will need to specify a shipGroup value (1, 2, 3, etc) for each shipping group you want fulfilled. See figure. **Important:** If no shipGroup value is specified, only the item associated with the first shipGroup (1) will be fulfilled.

This figure shows two different shipping groups: shipGroup 1 and shipGroup 2. When initializing the transaction, you must specify each item you want fulfilled based on its shipGroup value.

Does MSR work on custom forms?

Yes. However, after you enable Multiple Shipping Routes in your account, you must also enable MSR on your custom form by adding the Enable Line Item Shipping checkbox to the Items sublist. For steps on adding this checkbox to a custom form, see *Multiple Shipping Routes* in the NetSuite Help Center.

Note: After MSR is enabled in your account, the Enable Line Item Shipping check box is automatically added to the Items sublist on standard forms.

Multiple Shipping Routes Code Samples for Web Services

The following samples provide a general workflow for web services developers using MSR. These samples use sales order as the transaction type.

In this workflow you can:

1. Create a sales order and set your items

2. Get the sales order
3. Update the sales order items with MSR shipping information
4. Retrieve the updated shipGroup data and update the shipping cost
5. Update the sales order shipping group
6. Initialize the sales order to create the item fulfillment
7. Search for unfulfilled sales orders that are MSR-enabled

Create a sales order and set your items

This sample shows how to create a sales order and add items. Even if MSR is enabled in your NetSuite account, as this sample shows, you can continue to create sales orders that do not have MSR set for each items. MSR only becomes enabled on a transaction after *isMultiShipTo* is set to *true*.

Note: See the sample Update the sales order items with MSR shipping information for details on converting a non-MSR transaction to an MSR-enabled transaction.

C#

```
// First, initiate login
NetSuiteService nss = new NetSuiteService();

// provide login details....
// Create a sales order
SalesOrder so = new SalesOrder();

RecordRef customer = new RecordRef();
customer.internalId = "123";
so.entity = customer;

// Set the order status on the sales order
so.orderStatus = SalesOrderOrderStatus._pendingFulfillment;
so.orderStatusSpecified = true;

// Add items
SalesOrderItem [] items = new SalesOrderItem[2];
SalesOrderItemList itemList = new SalesOrderItemList();

SalesOrderItem item1 = new SalesOrderItem();
RecordRef item1Ref = new RecordRef();
item1Ref.internalId = "40";
item1.item = item1Ref;
item1.amountSpecified = true;
item1.amount = 3.0;

SalesOrderItem item2 = new SalesOrderItem();
RecordRef itemRef2 = new RecordRef();
itemRef2.internalId = "45";
item2.item = itemRef2;
item2.quantitySpecified = true;
item2.quantity = 2;

items[0] = item1;
items[1] = item2;
```

```
itemList.item = items;
so.itemList = itemList;

WriteResponse ws = nss.add(so);
```

SOAP Request

```
            <record xmlns:q1="urn:sales_2017_1.transactions.webservices.netsuite.com" xsi:type=
"q1:SalesOrder">
               <q1:entity internalId="123" />
          <q1:orderStatus>_pendingFulfillment</q1:orderStatus>
               <q1:itemList>
                  <q1:item>
                     <q1:item internalId="40" />
                     <q1:amount>3</q1:amount>
                  </q1:item>
                  <q1:item>
                     <q1:item internalId="45" />
                     <q1:quantity>2</q1:quantity>
                  </q1:item>
               </q1:itemList>
            </record>
```

SOAP Response

```
<writeResponse>
  <platformCore:status isSuccess="true"
   xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
       <baseRef internalId="993" type="salesOrder" xsi:type="platformCore:RecordRef"
      xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
</writeResponse>
```

Get the sales order

After adding items to a sales order, you can then get the sales order and enable MSR on the specific transaction. Note, you could have enabled MSR on the transaction when you originally created the sales order object. To do so, you needed to set *isMultiShipTo* to true in the first sample.

C#

```
RecordRef rr = new RecordRef();
rr = (RecordRef) ws.baseRef;
String sKey = rr.internalId;
ReadResponse getResp = nss.get(rr);
```

SOAP Request

```
          <get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
               <baseRef xmlns:q1="urn:core_2017_1.platform.webservices.netsuite.com" xsi:type="
q1:RecordRef"
      internalId="993" type="salesOrder" />
```

```
          </get>
```

SOAP Response

```
                 <record internalId="993" xsi:type="tranSales:SalesOrder"
        xmlns:tranSales="urn:sales_2017_1.transactions.webservices.netsuite.com">
                    <tranSales:createdDate>2008-09-15T17:29:00.000-07:00</tranSales:createdDat
e>
                    <tranSales:entity internalId="123"
        xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                       <platformCore:name>Carrie Davis</platformCore:name>
                    </tranSales:entity>
                    <tranSales:tranDate>2008-09-15T00:00:00.000-07:00</tranSales:tranDate>
                    <tranSales:tranId>183</tranSales:tranId>
                    <tranSales:source>web services</tranSales:source>
                    <tranSales:orderStatus>_pendingFulfillment</tranSales:orderStatus>
                    <tranSales:salesRep internalId="111"
        xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                       <platformCore:name>Corporate Sales Team</platformCore:name>
                    </tranSales:salesRep>
                    <tranSales:partner internalId="129"
        xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                       <platformCore:name>The Bay Times</platformCore:name>
                    </tranSales:partner>
                    <tranSales:leadSource internalId="100002"
        xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                       <platformCore:name>New Registration Promotion</platformCore:name>
                    </tranSales:leadSource>
                    <tranSales:salesEffectiveDate>2008-09-15T00:00:00.000-07:00</tranSales:sal
esEffectiveDate>
                    <tranSales:excludeCommission>false</tranSales:excludeCommission>
                    <tranSales:promoCode internalId="2"
        xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                       <platformCore:name>New Registration Promotion</platformCore:name>
                    </tranSales:promoCode>
                    <tranSales:toBePrinted>false</tranSales:toBePrinted>
                    <tranSales:toBeEmailed>false</tranSales:toBeEmailed>
                    <tranSales:email>carrie10101@freeversion.1org</tranSales:email>
                    <tranSales:toBeFaxed>false</tranSales:toBeFaxed>
                    <tranSales:transactionBillAddress
        xmlns:platformCommon="urn:common_2017_1.platform.webservices.netsuite.com">
                       <platformCommon:billAddr1>550504  Mission Street</platformCommon:billAd
dr1>
                       <platformCommon:billCity>Santa Cruz</platformCommon:billCity>
                       <platformCommon:billState>CA</platformCommon:billState>
                       <platformCommon:billZip>95060</platformCommon:billZip>
                       <platformCommon:billCountry>_unitedStates</platformCommon:billCountry>
                    </tranSales:transactionBillAddress>
                    <tranSales:billAddress>550504  Mission Street<br>Santa Cruz CA 95060<br>Un
ited
        States</tranSales:billAddress>
                    <tranSales:transactionShipAddress
        xmlns:platformCommon="urn:common_2017_1.platform.webservices.netsuite.com">
                       <platformCommon:shipAddr1>550504  Mission Street</platformCommon:shipAd
```

```
dr1>
                        <platformCommon:shipCity>Santa Cruz</platformCommon:shipCity>
                        <platformCommon:shipState>CA</platformCommon:shipState>
                        <platformCommon:shipZip>95060</platformCommon:shipZip>
                        <platformCommon:shipCountry>_unitedStates</platformCommon:shipCountry>
                        <platformCommon:shipIsResidential>true</platformCommon:shipIsResidentia
l>
                     </tranSales:transactionShipAddress>
                     <tranSales:shipAddress>550504  Mission Street<br>Santa Cruz CA 95060<br>Un
ited
         States</tranSales:shipAddress>
                     <tranSales:shipDate>2008-0911a4-15T00:00:00.000-07:00</tranSales:shipDate>

                     <tranSales:shipMethod internalId="3"
         xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                        <platformCore:name>Special Delivery</platformCore:name>
                     </tranSales:shipMethod>
                     <tranSales:shippingCost>5.0</tranSales:shippingCost>
                     <tranSales:isMultiShipTo>false</tranSales:isMultiShipTo>
                     <tranSales:shippingTaxCode internalId="-7"
         xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                        <platformCore:name>-Not Taxable-</platformCore:name>
                     </tranSales:shippingTaxCode>
                     <tranSales:handlingTaxCode internalId="-7"
         xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                        <platformCore:name>-Not Taxable-</platformCore:name>
                     </tranSales:handlingTaxCode>
                     <tranSales:handlingCost>6.0</tranSales:handlingCost>
                     <tranSales:shipComplete>false</tranSales:shipComplete>
                     <tranSales:ccApproved>false</tranSales:ccApproved>
                     <tranSales:altSalesTotal>0.0</tranSales:altSalesTotal>
                     <tranSales:subTotal>24.9</tranSales:subTotal>
                     <tranSales:altShippingCost>5.0</tranSales:altShippingCost>
                     <tranSales:altHandlingCost>6.0</tranSales:altHandlingCost>
                     <tranSales:total>35.9</tranSales:total>
                     <tranSales:subsidiary internalId="1"
         xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                        <platformCore:name>Parent Company</platformCore:name>
                     </tranSales:subsidiary>
                     <tranSales:lastModifiedDate>2008-09-15T17:29:00.000-07:00</tranSales:lastM
odifiedDate>
                     <tranSales:status>Pending Fulfillment</tranSales:status>
                     <tranSales:itemList>
                        <tranSales:item>
                           <tranSales:item internalId="40"
         xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                              <platformCore:name>Accessories : Crusher Game Pad</platformCore:n
ame>
                           </tranSales:item>
                           <tranSales:quantity>1.0</tranSales:quantity>
                           <tranSales:description>Crusher Game Pad</tranSales:description>
                           <tranSales:price internalId="1"
         xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                              <platformCore:name>Base Price</platformCore:name>
                           </tranSales:price>
                           <tranSales:rate>12.95</tranSales:rate>
                           <tranSales:amount>3.0</tranSales:amount>
                           <tranSales:commitInventory>_availableQty</tranSales:commitInventory>

                           <tranSales:isClosed>false</tranSales:isClosed>
                           <tranSales:fromJob>false</tranSales:fromJob>
                           <tranSales:isEstimate>false</tranSales:isEstimate>
                           <tranSales:line>1</tranSales:line>
                           <tranSales:quantityBackOrdered>0.0</tranSales:quantityBackOrdered>
                           <tranSales:quantityBilled>0.0</tranSales:quantityBilled>
                           <tranSales:quantityCommitted>1.0</tranSales:quantityCommitted>
                           <tranSales:quantityFulfilled>0.0</tranSales:quantityFulfilled>
                           <tranSales:taxCode internalId="-7"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                              <platformCore:name>-Not Taxable-</platformCore:name>
                           </tranSales:taxCode>
                        </tranSales:item>
                        <tranSales:item>
                           <tranSales:item internalId="45"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                              <platformCore:name>Accessories : Cable - Cat 5, 10 ft</platformCo
re:name>
                           </tranSales:item>
                           <tranSales:quantity>2.0</tranSales:quantity>
                           <tranSales:description>Cat 5 Patch Cable 10 ft</tranSales:descriptio
n>
                           <tranSales:price internalId="1"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                              <platformCore:name>Base Price</platformCore:name>
                           </tranSales:price>
                           <tranSales:rate>10.95</tranSales:rate>
                           <tranSales:amount>21.9</tranSales:amount>
                           <tranSales:commitInventory>_availableQty</tranSales:commitInventory>

                           <tranSales:isClosed>false</tranSales:isClosed>
                           <tranSales:fromJob>false</tranSales:fromJob>
                           <tranSales:isEstimate>false</tranSales:isEstimate>
                           <tranSales:line>2</tranSales:line>
                           <tranSales:quantityBackOrdered>0.0</tranSales:quantityBackOrdered>
                           <tranSales:quantityBilled>0.0</tranSales:quantityBilled>
                           <tranSales:quantityCommitted>2.0</tranSales:quantityCommitted>
                           <tranSales:quantityFulfilled>0.0</tranSales:quantityFulfilled>
                           <tranSales:taxCode internalId="-7"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                              <platformCore:name>-Not Taxable-</platformCore:name>
                           </tranSales:taxCode>
                        </tranSales:item>
                     </tranSales:itemList>
                  <tranSales:customFieldList
         xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                        <platformCore:customField internalId="215" scriptId="custbody1"
         xsi:type="platformCore:StringCustomFieldRef">
                           <platformCore:value>800-555-9177</platformCore:value>
                        </platformCore:customField>
                     </tranSales:customFieldList>
```

```
                </record>
```

Update the sales order items with MSR shipping information

This sample shows how to convert a sales order that does not yet have MSR enabled to a sales order that does. To do this you must set *isMulitShipTo* to *true* and then specify the *shipAddress* and *shipMethod* for each item.

C#

```
// Keep a record of the sales order from previous query operation
SalesOrder soCreated = (SalesOrder) getResp.record;
SalesOrder soUpdate = new SalesOrder();
soUpdate.internalId = soCreated.internalId;

// Enable line level shipping. This is equivalent to selecting
// the Enable Line Level Shipping checkbox on the Items tab.
soUpdate.isMultiShipToSpecified = true;
soUpdate.isMultiShipTo = true;
// set shipMethods and shipAddresses
SalesOrderItem [] itemsUpdate = new SalesOrderItem[soCreated.itemList.item.Length];
SalesOrderItem item1Update = new SalesOrderItem();
item1Update.lineSpecified=true;
item1Update.line = soCreated.itemList.item[0].line;
RecordRef shipAddr1 = new RecordRef();
shipAddr1.internalId = "93";
item1Update.shipAddress = shipAddr1;

RecordRef shipMeth1 = new RecordRef();
shipMeth1.internalId = "2";
item1Update.shipMethod = shipMeth1;
SalesOrderItem item2Update = new SalesOrderItem();
item2Update.lineSpecified=true;
item2Update.line = soCreated.itemList.item[1].line;
RecordRef shipAddr2 = new RecordRef();
shipAddr2.internalId = "244715";
item2Update.shipAddress = shipAddr2;
RecordRef shipMeth2 = new RecordRef();
shipMeth2.internalId = "92";
item2Update.shipMethod = shipMeth2;
itemsUpdate[0] = item1Update;
itemsUpdate[1] = item2Update;
SalesOrderItemList itemListUpdate = new SalesOrderItemList();
itemListUpdate.item = itemsUpdate;
soUpdate.itemList = itemListUpdate;
nss.update(soUpdate);
```

SOAP Request

```
                <update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
                   <record xmlns:q1="urn:sales_2017_1.transactions.webservices.netsuite.com"
          xsi:type="q1:SalesOrder" internalId="997">
                      <q1:isMultiShipTo>true</q1:isMultiShipTo>
```

```
                    <q1:itemList>
                       <q1:item>
                          <q1:line>1</q1:line>
                          <q1:shipAddress internalId="93" />
                          <q1:shipMethod internalId="2" />
                       </q1:item>
                       <q1:item>
                          <q1:line>2</q1:line>
                          <q1:shipAddress internalId="244715" />
                          <q1:shipMethod internalId="92" />
                       </q1:item>
                    </q1:itemList>
                 </record>
              </update>
```

SOAP Response

```
<writeResponse>
  <platformCore:status isSuccess="true"
   xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
                    <baseRef internalId="997" type="salesOrder" xsi:type="platformCore:RecordR
ef"
         xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
</writeResponse>
```

Retrieve the updated shipGroup data and update the shipping cost

If you choose, you can get the sales order and then specify which item you want to update based on the item's shipGroup value.

C#

```
// Retrieve the updated shipGroup data and then update
// the shipping cost
ReadResponse getRespUpdated = nss.get(rr);
```

SOAP Request

```
<get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <baseRef xmlns:q1="urn:core_2017_1.platform.webservices.netsuite.com" xsi:type="q1:RecordRef
"
   internalId="997" type="salesOrder" />
</get>
```

SOAP Response

```
<record internalId="997" xsi:type="tranSales:SalesOrder"
   xmlns:tranSales="urn:sales_2017_1.transactions.webservices.netsuite.com">
               <tranSales:createdDate>2008-09-16T10:51:00.000-07:00</tranSales:createdDate>
              <tranSales:entity internalId="123"
```

```
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                              <platformCore:name>Carrie Davis</platformCore:name>
                           </tranSales:entity>
                           <tranSales:tranDate>2008-09-16T00:00:00.000-07:00</tranSales:tranDat
e>
                           <tranSales:tranId>186</tranSales:tranId>
                           <tranSales:source>web services</tranSales:source>
                           <tranSales:orderStatus>_pendingFulfillment</tranSales:orderStatus>
                           <tranSales:salesRep internalId="111"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                              <platformCore:name>Corporate Sales Team</platformCore:name>
                           </tranSales:salesRep>
                           <tranSales:partner internalId="129"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                              <platformCore:name>The Bay Times</platformCore:name>
                           </tranSales:partner>
                           <tranSales:leadSource internalId="100002"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                              <platformCore:name>New Registration Promotion</platformCore:name>

                           </tranSales:leadSource>
                           <tranSales:salesEffectiveDate>2008-09-16T00:00:00.000-07:00</tranSal
es:salesEffectiveDate>
                           <tranSales:excludeCommission>false</tranSales:excludeCommission>
                           <tranSales:promoCode internalId="2"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                              <platformCore:name>New Registration Promotion</platformCore:name>

                           </tranSales:promoCode>
                           <tranSales:toBePrinted>false</tranSales:toBePrinted>
                           <tranSales:toBeEmailed>false</tranSales:toBeEmailed>
                           <tranSales:email>carrie10101@freeversion.1org</tranSales:email>
                           <tranSales:toBeFaxed>false</tranSales:toBeFaxed>
                           <tranSales:transactionBillAddress
            xmlns:platformCommon="urn:common_2017_1.platform.webservices.netsuite.com">
                              <platformCommon:billAddr1>550504  Mission Street</platformCommon:
billAddr1>
                              <platformCommon:billCity>Santa Cruz</platformCommon:billCity>
                              <platformCommon:billState>CA</platformCommon:billState>
                              <platformCommon:billZip>95060</platformCommon:billZip>
                              <platformCommon:billCountry>_unitedStates</platformCommon:billCou
ntry>
                           </tranSales:transactionBillAddress>
                           <tranSales:billAddress>550504  Mission Street<br>Santa Cruz CA 95060
<br>United
            States</tranSales:billAddress>
                           <tranSales:transactionShipAddress
            xmlns:platformCommon="urn:common_2017_1.platform.webservices.netsuite.com">
                              <platformCommon:shipIsResidential>false</platformCommon:shipIsRes
idential>
                           </tranSales:transactionShipAddress>
                           <tranSales:shipDate>2008-09-16T00:00:00.000-07:00</tranSales:shipDat
e>
                           <tranSales:isMultiShipTo>true</tranSales:isMultiShipTo>
                           <tranSales:shipComplete>false</tranSales:shipComplete>
```

```
                            <tranSales:ccApproved>false</tranSales:ccApproved>
                            <tranSales:altSalesTotal>0.0</tranSales:altSalesTotal>
                            <tranSales:subTotal>24.9</tranSales:subTotal>
                            <tranSales:altShippingCost>11.87</tranSales:altShippingCost>
                            <tranSales:altHandlingCost>4.0</tranSales:altHandlingCost>
                            <tranSal11a4es:total>40.77</tranSales:total>
                            <tranSales:subsidiary internalId="1"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                               <platformCore:name>Parent Company</platformCore:name>
                            </tranSales:subsidiary>
                            <tranSales:lastModifiedDate>2008-09-16T10:51:00.000-07:00</tranSales
:lastModifiedDate>
                            <tranSales:status>Pending Fulfillment</tranSales:status>
                            <tranSales:itemList>
                               <tranSales:item>
                                  <tranSales:item internalId="40"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                     <platformCore:name>Accessories : Crusher Game Pad</platform
Core:name>
                                  </tranSales:item>
                                  <tranSales:quantity>1.0</tranSales:quantity>
                                  <tranSales:description>Crusher Game Pad</tranSales:description
>
                                  <tranSales:price internalId="1"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                     <platformCore:name>Base Price</platformCore:name>
                                  </tranSales:price>
                                  <tranSales:rate>12.95</tranSales:rate>
                                  <tranSales:amount>3.0</tranSales:amount>
                                  <tranSales:commitInventory>_availableQty</tranSales:commitInve
ntory>
                                  <tranSales:isClosed>false</tranSales:isClosed>
                                  <tranSales:fromJob>false</tranSales:fromJob>
                                  <tranSales:isEstimate>false</tranSales:isEstimate>
                                  <tranSales:line>1</tranSales:line>
                                  <tranSales:quantityBackOrdered>0.0</tranSales:quantityBackOrde
red>
                                  <tranSales:quantityBilled>0.0</tranSales:quantityBilled>
                                  <tranSales:quantityCommitted>1.0</tranSales:quantityCommitted>

                                  <tranSales:quantityFulfilled>0.0</tranSales:quantityFulfilled>

                                  <tranSales:taxCode internalId="-7"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                     <platformCore:name>-Not Taxable-</platformCore:name>
                                  </tranSales:taxCode>
                                  <tranSales:shipAddress internalId="93"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                     <platformCore:name>Billing Address</platformCore:name>
                                  </tranSales:shipAddress>
                                  <tranSales:shipMethod internalId="2"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                     <platformCore:name>Pick-up</platformCore:name>
                                  </tranSales:shipMethod>
                               </tranSales:item>
```

```
                              <tranSales:item>
                                 <tranSales:item internalId="45"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                    <platformCore:name>Accessories : Cable - Cat 5, 10 ft</plat
formCore:name>
                                 </tranSales:item>
                                 <tranSales:quantity>2.0</tranSales:quantity>
                                 <tranSales:description>Cat 5 Patch Cable 10 ft</tranSales:desc
ription>
                                 <tranSales:price internalId="1"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                    <platformCore:name>Base Price</platformCore:name>
                                 </tranSales:price>
                                 <tranSales:rate>10.95</tranSales:rate>
                                 <tranSales:amount>21.9</tranSales:amount>
                                 <tranSales:commitInventory>_availableQty</tranSales:commitInve
ntory>
                                 <tranSales:isClosed>false</tranSales:isClosed>
                                 <tranSales:fromJob>false</tranSales:fromJob>
                                 <tranSales:isEstimate>false</tranSales:isEstimate>
                                 <tranSales:line>2</tranSales:line>
                                 <tranSales:quantityBackOrdered>0.0</tranSales:quantityBackOrde
red>
                                 <tranSales:quantityBilled>0.0</tranSales:quantityBilled>
                                 <tranSales:quantityCommitted>2.0</tranSales:quantityCommitted>

                                 <tranSales:quantityFulfilled>0.0</tranSales:quantityFulfilled>

                                 <tranSales:taxCode internalId="-7"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                    <platformCore:name>-Not Taxable-</platformCore:name>
                                 </tranSales:taxCode>
                                 <tranSales:shipAddress internalId="244715"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                    <platformCore:name>san mateo shipping</platformCore:name>
                                 </tranSales:shipAddress>
                                 <tranSales:shipMethod internalId="92"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                    <platformCore:name>UPS Ground</platformCore:name>
                                 </tranSales:shipMethod>
                              </tranSales:item>
                           </tranSales:itemList>
                           <tranSales:shipGroupList>
                              <tranSales:shipGroup>
                                 <tranSales:id>1</tranSales:id>
                                 <tranSales:isFulfilled>false</tranSales:isFulfilled>
                                 <tranSales:weight>1.0</tranSales:weight>
                                 <tranSales:sourceAddressRef internalId="DEFAULT"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
                                 <tranSales:sourceAddress>1500 3rd St San Mateo, CA 94403 US</t
ranSales:sourceAddress>
                                 <tranSales:destinationAddressRef internalId="93"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
                                 <tranSales:destinationAddress>CO United States</tranSales:dest
inationAddress>
```

```
                                 <tranSales:shippingMethodRef internalId="2"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
                                 <tranSales:shippingMethod>Pick-up</tranSales:shippingMethod>
                                 <tranSales:handlingRate>0.0</tranSales:handlingRate>
                                 <tranSales:shippingRate>5.0</tranSales:shippingRate>
                              </tranSales:shipGroup>
                              <tra50anSales:shipGroup>
                                 <tranSales:id>2</tranSales:id>
                                 <tranSales:isFulfilled>false</tranSales:isFulfilled>
                                 <tranSales:weight>2.0</tranSales:weight>
                                 <tranSales:sourceAddressRef internalId="DEFAULT"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
                                 <tranSales:sourceAddress>1500 3rd St San Mateo, CA 94403 US</t
ranSales:sourceAddress>
                                 <tranSales:destinationAddressRef internalId="244715"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
                                 <tranSales:destinationAddress>2955 Campus Drive 175 San Mateo
CA 94403 United
            States</tranSales:destinationAddress>
                                 <tranSales:shippingMethodRef internalId="92"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
                                 <tranSales:shippingMethod>UPS Ground</tranSales:shippingMethod
>

                                 <tranSales:handlingRate>4.0</tranSales:handlingRate>
                                 <tranSales:shippingRate>6.87</tranSales:shippingRate>
                              </tranSales:shipGroup>
                           </tranSales:shipGroupList>
                           <tranSales:customFieldList
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                              <platformCore:customField internalId="215" scriptId="custbody1"
            xsi:type="platformCore:StringCustomFieldRef">
                                 <platformCore:value>800-555-9177</platformCore:value>
                              </platformCore:customField>
                           </tranSales:customFieldList>
                        </record>
```

Update the sales order shipping group

By specifying a shipping group, you can then update the data associated with that shipping group. This data includes item shipping and handling rates.

C#

```
// Update the sales order shipping groups, for example, the shipping rate

// Keep a record of the SO updated
SalesOrder soUpdated = (SalesOrder) getRespUpdated.record;
SalesOrder soShipGroupUpdate = new SalesOrder();
soShipGroupUpdate.internalId = soUpdated.internalId;

TransactionShipGroup shipGroup1 = new TransactionShipGroup();
shipGroup1.idSpecified = true;
shipGroup1.id = soUpdated.shipGroupList.shipGroup[0].id;
shipGroup1.shippingRateSpecified = true;
```

```
shipGroup1.shippingRate = 10.0;
TransactionShipGroup [] shipGroups = new TransactionShipGroup[1];
shipGroups[0] = shipGroup1;

SalesOrderShipGroupList shipGroupList = new SalesOrderShipGroupList();
shipGroupList.shipGroup = shipGroups;
soShipGroupUpdate.shipGroupList = shipGroupList;
nss.update(soShipGroupUpdate);
```

SOAP Request

```
<update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <record xmlns:q1="urn:sales_2017_1.transactions.webservices.netsuite.com" xsi:type="q1:SalesO
rder"
   internalId="997">
   <q1:shipGroupList>
      <q1:shipGroup>
         <q1:id>1</q1:id>
         <q1:shippingRate>10</q1:shippingRate>
     </q1:shipGroup>
 </q1:shipGroupList>
 </record>
</update>
```

SOAP Response

```
<writeResponse>
   <platformCore:status isSuccess="true"
   xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
          <baseRef internalId="997" type="salesOrder" xsi:type="platformCore:RecordRef"
        xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
```

Initialize the sales order to create the item fulfillment

Next, you can run initialize() to obtain the item fulfillment record.

Note: The initialize operation includes an optional argument called *AuxReference*. In *AuxReference* users can set the type to *shippingGroup* by using the enum object InitializeAuxRefType.

When working with MSR-enabled transactions, you must specify a value for *shippingGroup*. If you do not specify a value, the value **1** (for the first shipping group) is defaulted. This means that only the items belonging to the first shipping group will be fulfilled when the sales order is initialized.

C#

```
for (int j=0; j<soUpdated.shipGroupList.shipGroup.Length; j++)
{
   InitializeRecord ir = new InitializeRecord();
   ir.type = InitializeType.itemFulfillment;
   InitializeRef iref = new InitializeRef();
   iref.typeSpecified=true;
   iref.type=InitializeRefType.salesOrder;
   iref.internalId = rr.internalId;
   ir.reference = iref;
   InitializeAuxRef iar = new InitializeAuxRef();
   iar.typeSpecified=true;
   iar.type = InitializeAuxRefType.shippingGroup;
   iar.internalId = soUpdated.shipGroupList.shipGroup[j].id.ToString();
   ir.auxReference = iar;
   ReadResponse getInitResp = nss.initialize(ir);
      if (getInitResp.status.isSuccess)
      {
      // Keep a record of the original item fulfillment
      ItemFulfillment ifrec = (ItemFulfillment) getInitResp.record;

      //SalesOrderShipGroupList sgListForIF = ifrec..shipGroupList;
      //TransactionShipGroup [] shipGroupsForIF = sgListForIF.shipGroup;

      ItemFulfillment recToFulfill = new ItemFulfillment();
      // Set createdFrom field
      recToFulfill.createdFrom = ifrec.createdFrom;
      // Set createdFromShipGroup field for multiple shipping routes enabled orders
      recToFulfill.createdFromShipGroupSpecified=true;
      recToFulfill.createdFromShipGroup = soUpdated.shipGroupList.shipGroup[j].id;
      ItemFulfillmentItemList ifitemlist = ifrec.itemList;
      ItemFulfillmentItem [] ifitems = new ItemFulfillmentItem[ifitemlist.item.Length];
      RecordRef locRef = new RecordRef();
      locRef.internalId = "1";
         for (int i=0; i<ifitemlist.item.Length; i++)
         {
         ItemFulfillmentItem ffItem = new ItemFulfillmentItem();
         ffItem.item = ifitemlist.item[i].item;
         ffItem.orderLineSpecified=true;
         ffItem.orderLine = ifitemlist.item[i].orderLine;
         ffItem.location = locRef;
         ifitems[i] = ffItem;
         }
         ItemFulfillmentItemList ifitemlistToFulfill = new ItemFulfillmentItemList();
         ifitemlistToFulfill.item = ifitems;
         recToFulfill.itemList = ifitemlistToFulfill;
         nss.add(recToFulfill);
         }
   }
```

SOAP Request

```
                        <initialize xmlns="urn:messages_2017_1.platform.webservices.netsuite.co
m">
                           <initializeRecord>
                              <type xmlns="urn:core_2017_1.platform.webservices.netsuite.com">i
temFulfillment</type>
                              <reference type="salesOrder" internalId="997"
            xmlns="urn:core_2017_1.platform.webservices.netsuite.com" />
                              <auxReference type="shippingGroup" internalId="1"
            xmlns="urn:core_2017_1.platform.webservices.netsuite.com" />
```

```
                        </initializeRecord>
                     </initialize>
```

SOAP Response

```
                     <initializeResponse xmlns="urn:messages_2017_1.platform.webservices.net
suite.com">
                        <readResponse>
                           <platformCore:status isSuccess="true"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
                           <record xsi:type="tranSales:ItemFulfillment"
            xmlns:tranSales="urn:sales_2017_1.transactions.webservices.netsuite.com">
                              <tranSales:createdDate>2008-09-16T10:51:00.000-07:00</tranSale
s:createdDate>
                              <tranSales:lastModifiedDate>2008-09-16T10:51:00.000-07:00</tra
nSales:lastModifiedDate>
                              <tranSales:postingPeriod internalId="132"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                 <platformCore:name>Sep 2008</platformCore:name>
                              </tranSales:postingPeriod>
                              <tranSales:entity internalId="123"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                 <platformCore:name>Carrie Davis</platformCore:name>
                              </tranSales:entity>
                              <tranSales:createdFrom internalId="997"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                 <platformCore:name>Sales Order #186</platformCore:name>
                              </tranSales:createdFrom>
                              <tranSales:transactionShipAddress
            xmlns:platformCommon="urn:common_2017_1.platform.webservices.netsuite.com">
                                 <platformCommon:shipAttention>Billing Address</platformComm
on:shipAttention>
                                 <platformCommon:shipAddressee>Carrie Davis</platformCommon:
shipAddressee>
                                 <platformCommon:shipPhone>800-555-9177</platformCommon:ship
Phone>
                                 <platformCommon:shipState>CO</platformCommon:shipState>
                                 <platformCommon:shipCountry>_unitedStates</platformCommon:s
hipCountry>
                                 <platformCommon:shipIsResidential>false</platformCommon:shi
pIsResidential>
                              </tranSales:transactionShipAddress>
                              <tranSales:shipAddressList internalId="93"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                 <platformCore:name>Billing Address</platformCore:name>
                              </tranSales:shipAddressList>
                              <tranSales:shipAddress>Carrie DavisCO US</tranSales:shipAddres
s>
                              <tranSales:tranDate>2008-09-16T00:00:00.000-07:00</tranSales:t
ranDate>
                              <tranSales:tranId>39</tranSales:tranId>
                              <tranSales:shipMethod internalId="2"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                 <platformCore:name>Pick-up</platformCore:name>
```

```
                                  </tranSales:shipMethod>
                                  <tranSales:shippingCost>10.0</tranSales:shippingCost>
                                  <tranSales:handlingCost>0.0</tranSales:handlingCost>
                                  <tranSales:itemList>
                                     <tranSales:item>
                                        <tranSales:itemReceive>true</tranSales:itemReceive>
                                        <tranSales:itemName>Crusher Game Pad </tranSales:itemNam
e>
                                        <tranSales:description>Crusher Game Pad</tranSales:descr
iption>
                                        <tranSales:quantity>1.0</tranSales:quantity>
                                        <tranSales:item internalId="40"
               xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                                           <platformCore:name>Crusher Game Pad</platformCore:nam
e>
                                        </tranSales:item>
                                        <tranSales:orderLine>1</tranSales:orderLine>
                                        <tranSales:quantityRemaining>1.0</tranSales:quantityRema
ining>
                                     </tranSales:item>
                                  </tranSales:itemList>
                               </record>
                            </readResponse>
                         </initializeResponse>
```

SOAP Request (Item Fulfillment)

```
                            <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
                               <record xmlns:q1="urn:sales_2017_1.transactions.webservices.netsu
ite.com"
            xsi:type="q1:ItemFulfillment">
                                  <q1:createdFrom internalId="997">
                                     <name xmlns="urn:core_2017_1.platform.webservices.netsuite.
com">Sales Order #186
               </name>
                                  </q1:createdFrom>
                                  <q1:itemList>
                                     <q1:item>
                                        <q1:location internalId="1" />
                                        <q1:item internalId="40">
                                           <name xmlns="urn:core_2017_1.platform.webservices.net
suite.com">Crusher Game
               Pad</name>
                                        </q1:item>
                                        <q1:orderLine>1</q1:orderLine>
                                     </q1:item>
                                  </q1:itemList>
                               </record>
                            </add>
```

SOAP Response (Item Fulfillment)

```
<writeResponse>
```

```
   <platformCore:status isSuccess="true"
     xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
    <baseRef internalId="998" type="itemFulfillment" xsi:type="platformCore:RecordRef"
    xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
</writeResponse>
```

Search for unfulfilled sales orders that are MSR-enabled

This sample shows how to search for unfulfilled sales orders that are MSR-enabled. For details on how to specify additional search criteria, see the help topic Advanced Searches in Web Services.

C#

```
  // Define search criteria and return columns
 TransactionSearchAdvanced tsa4 = new TransactionSearchAdvanced();

// Set search preference to return search columns
SearchPreferences sp = new SearchPreferences();
sp.returnSearchColumns = true;
nss.searchPreferences = sp;
// Instantiate SearchRow 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;
// return item
SearchColumnSelectField [] itemIdCols = new SearchColumnSelectField[1];
SearchColumnSelectField itemId = new SearchColumnSelectField();
itemIdCols[0] = itemId;
tsrb.item = itemIdCols;
// return item fulfillment status
SearchColumnBooleanField [] isFulfilledCols = new SearchColumnBooleanField[1];
SearchColumnBooleanField isFulfilledCol = new SearchColumnBooleanField();
isFulfilledCol.customLabel = "Order Fulfilled";
isFulfilledCols[0] = isFulfilledCol;
tsrb.shipRecvStatusLine = isFulfilledCols;
// return tranDate
SearchColumnDateField [] tranDateCols = new SearchColumnDateField[1];
SearchColumnDateField tranDateCol = new SearchColumnDateField();
tranDateCols[0] = tranDateCol;
tsrb.tranDate = tranDateCols;
// return tranId
SearchColumnStringField [] tranIdCols = new SearchColumnStringField[1];
SearchColumnStringField tranIdCol = new SearchColumnStringField();
tranIdCols[0] = tranIdCol;
tsrb.tranId = tranIdCols;

tsr.basic = tsrb;
// Define search criteria
TransactionSearch ts = new TransactionSearch();
```

```
TransactionSearchBasic tsb = new TransactionSearchBasic();
// 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;
// tranId contains 183
SearchStringField sfTranId = new SearchStringField();
sfTranId.searchValue = "183";
sfTranId.@operator = SearchStringFieldOperator.contains;
sfTranId.operatorSpecified = true;
tsb.tranId=sfTranId;
// SO item unfulfilled
SearchBooleanField sbfTranStatus = new SearchBooleanField();
sbfTranStatus.searchValue = false;
sbfTranStatus.searchValueSpecified = true;
tsb.shipRecvStatusLine = sbfTranStatus;
ts.basic = tsb;
tsa4.criteria = ts;
tsa4.columns = tsr;
nss.search(tsa4);
```

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">
                 <q1:criteria>
                    <q1:basic>
                       <shipRecvStatusLine xmlns="urn:common_2017_1.platform.webservices.netsu
ite.com">
                          <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.co
m">false
            </searchValue>
                       </shipRecvStatusLine>
                       <tranId operator="contains" xmlns="urn:common_2017_1.platform.webservic
es.netsuite.com">
                          <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.co
m">183
            </searchValue>
                       </tranId>
                       <type operator="anyOf" xmlns="urn:common_2017_1.platform.webservices.ne
tsuite.com">
                          <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.co
m">_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.co
m">Sales Order ID
            </customLabel>
                         </internalId>
                         <item xmlns="urn:common_2017_1.platform.webservices.netsuite.com" />
                         <shipRecvStatusLine xmlns="urn:common_2017_1.platform.webservices.netsu
ite.com">
                            <customLabel xmlns="urn:core_2017_1.platform.webservices.netsuite.co
m">Order Item
            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>
             </search>
          </soap:Body>
```

SOAP Response

```
<searchResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
               <platformCore:searchResult
         xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                  <platformCore:status isSuccess="true"/>
                  <platformCore:totalRecords>1</platformCore:totalRecords>
                  <platformCore:pageSize>10</platformCore:pageSize>
                  <platformCore:totalPages>1</platformCore:totalPages>
                  <platformCore:pageIndex>1</platformCore:pageIndex>         <platformCore:sear
chId>WEBSERVICES_MSTRWLF_11142008164657795950123256l_cb6492913ad1</platformCore:searchId>
                  <platformCore:searchRowList>
                     <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="987"/>
                              <platformCore:customLabel>Sales Order ID</platformCore:customLabe
l>
                           </platformCommon:internalId>
                           <platformCommon:item>
                              <platformCore:searchValue internalId="40"/>
                           </platformCommon:item>
                           <platformCommon:shipRecvStatusLine>
                              <platformCore:searchValue>false</platformCore:searchValue>
                              <platformCore:customLabel>Order Item Fulfilled</platformCore:cust
omLabel>
                           </platformCommon:shipRecvStatusLine>
                           <platformCommon:tranDate>
```

```
                              <platformCore:searchValue>2008-11-14T00:00:00.000-08:00</platform
Core:searchValue>
                           </platformCommon:tranDate>
                           <platformCommon:tranId>
                              <platformCore:searchValue>183</platformCore:searchValue>
                           </platformCommon:tranId>
                        </tranSales:basic>
                     </platformCore:searchRow>
                  </platformCore:searchRowList>
               </platformCore:searchResult>
            </searchResponse>
```

Items

Review the following for general information about SuiteTalk items:

- Usage Notes for Item Record Types
- Working with Matrix Items in Web Services
- Shared Field Definitions for Items

The following item record types are supported in SuiteTalk:

- Assembly Item (BOM Item)
- Description Item
- Discount Item
- Download Item
- Gift Certificate Item
- Inventory Item
- Item Group
- Kit/Package Item
- Lot Numbered Assembly Item
- Lot Numbered Inventory Item
- Markup Item
- Noninventory Purchase Item
- Noninventory Resale Item
- NonInventory Sales Item
- Other Charge Purchase Item
- Other Charge Resale Item
- Other Charge Sale Item
- Payment Item
- Serialized Assembly Item
- Serialized Inventory Item
- Service Purchase Item
- Service Resale Item
- Service Sale Item
- Serialized Assembly Item

Usage Notes for Item Record Types

Note the following when working with item record types:

- Items are the goods and services you sell to customers, and the parts and raw materials you purchase from vendors. They can also include line items on sales and purchase forms, such as discounts and miscellaneous charges.
- Depending on the features enabled for the account specified during the login call, some item types may not be available.
- Setting the item base price will auto-calculate other pricing levels. For more information on setting item prices, see the help topic Item Pricing in the NetSuite Help Center.
- To make an item featured, you must set the item category to the Home tab in the NetSuite UI.

- The countryofmanufacture field for an item must be a country covered by FedEx or UPS.
- All item records are defined in the listAcct (accounting) XSD.

Working with Costing Methods

As of the 2013.1 endpoint, the costingMethod field was added to Lot Numbered Assembly Item, Lot Numbered Inventory Item, Serialized Assembly Item, and Serialized Inventory Item. This field is not mandatory for the add operation. Its value defaults to lot numbered for lot numbered items, and to specific for serialized items.

This field becomes read-only after it is set, so it is only used for add and advanced search operations. Note that this field is not returned in get and basic search operations. Instead, the read-only field costingMethodDisplay is returned. When this field is used as a result column in advanced search operations, it returns different values for the 2013.1 endpoint than in earlier endpoints. Before, the return value for lot numbered items was fifo, and the returned value for assembly items was lifo. Now the return value for lot numbered items is lot numbered and the return value for serialized items is specific.

The 2013.1 endpoint merged the CostingMethod enumeration in common.xsd with the ItemCostingMethod in accounting.xsd, so that both of these enumerations include the same values. CostingMethod is no longer necessary; ItemCostingMethod should be used.

```
<simpleType name="ItemCostingMethod">
       <restriction base="xsd:string">
            <enumeration value="_average"/>
            <enumeration value="_fifo"/>
            <enumeration value="_lifo"/>
            <enumeration value="_lotNumbered"/>
            <enumeration value="_serialized"/>
            <enumeration value="_standard"/>
        </restriction>
    </simpleType>
```

In a basic item search (complex type ItemSearchBasic in common.xsd), the criterion field costingMethod used the CostingMethod enumeration in earlier endpoints. In the 2013.1 endpoint and later, it uses the ItemCostingMethod enumeration. This enumeration is encapsulated by SearchEnumMultiSelectField:

```
 <element name="costingMethod" type="platformCore:SearchEnumMultiSelectField" minOccurs="0"/>
```

Working with Matrix Items in Web Services

Before working with matrix items in web services, you must first enable the Matrix Item feature in your NetSuite account. To enable this feature go to Setup > Company > Enable Features, on the Items & Inventory subtab, select the Matrix Items check box.

This section covers the following topics:

- Item Types that Support Matrix Options
- Creating a Parent Matrix Item
- Creating a Child Matrix Item
- Specifying Matrix Options
- Matrix Dimensions

- Updating Child and Parent Matrix Items
- Deleting Child and Parent Matrix Items
- Matrix Item Code Sample

Note: For general information about the matrix items feature, see the help topic Matrix Items.

Item Types that Support Matrix Options

You can use web services to create matrix items for the following record types:

- Inventory Item
- Lot Numbered Inventory Item
- Serialized Inventory Item
- Non-Inventory Item for Sale
- Non-Inventory Item for Resale
- Non-Inventory Item for Purchase
- Service Item for Sale
- Service Item for Resale
- Service Item for Purchase
- Other Charge Item for Sale
- Other Charge Item for Resale
- Other Charge Item for Purchase

Note that item types that supports matrix options includes the following elements:

- A matrixType enum field which can take the value of either _parent or _child to indicate if the submitted record is a parent or child matrix item. See Creating a Parent Matrix Item.
- A parent (Subitem of) field that points to a matrix parent item and populates the MatrixOption list. See Creating a Child Matrix Item.
- A MatrixOptionList element (similar to the Matrix tab on the UI). This element is an array of customFieldList of type select to populate matrix options on the child record. Users can specify only one value per matrix option field.

Creating a Parent Matrix Item

Users must set the matrixType field if they want to specify an item as either a parent or child matrix item.

Note: This is a deviation from the UI behavior. In the UI you cannot create a parent matrix item without selecting the child options, however, in web services this is possible.

Creating a Child Matrix Item

Assuming that users have already setup matrix item options in their account, they can proceed with creating child matrix items. As opposed to the UI where selecting matrix options on the parent will result in creating the matrix children, in web services the user has to add each child matrix item separately.

The user must set the RecordRef of the parent (Subitem of) field and specify one or more matrix options for the matrix child item record. Note that the parent item must have the is MatrixParent field set to true, otherwise the following error is thrown:

Error code = INVALID_MATRIX_PARENT

Error message = Item {1} is not a parent matrix item.

Users can set also set the externalId field to submit parent matrix item data and associate the child matrix items with the parent in a single addList request in web services.

Important: A maximum of 2000 children are permitted for a parent item.

Specifying Matrix Options

A call to getCustomization(“itemOptionCustomField”) can be used to return valid matrix option fields and values. A get on the parent matrix item returns only the matrix options that have been used. Not all possible options are returned.

Matrix Dimensions

After the first child is created, the dimensions for the matrix item are set and can not be changed. For example, if users creates a large, blue dress as the first child matrix item, they cannot subsequently create a dress that has any property other than size and color. A small, red, cotton dress will not be a valid option.

If users try to add a new child matrix with options that already exists for the parent, they will receive an error.

Updating Child and Parent Matrix Items

Users cannot update child matrix items through the parent record. Users must submit updates for child items individually.

For example, in web services, updating parent item values for the Asset Account, Cost of Goods Sold (COGS) account, or Income Account fields does not automatically update child item field values. You need to set these values for each child item row. The system does not enforce that child item values match the parent item values for these fields, so you need to ensure that they match yourself.

Also note that the following error is thrown if users attempt to update an existing item to make it a parent.

```
Error Code = USER_ERROR
Error Message = You can not change an existing item to make it a parent matrix item {1}.
```

Note: The values that are returned in MatrixOptionList are read-only after the child matrix item is added.

Deleting Child and Parent Matrix Items

In web services users cannot delete a parent item if it has children. They must delete the child items individually before deleting the parent.

A check for child matrix items occurs on a request to delete a parent matrix item. If no children exist, the parent item is deleted. If children do exist, an error is returned stating that the parent has child items.

Matrix Item Code Sample

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>
         <addList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record externalId="parentSweater" xsi:type="ns1:InventoryItem" xmlns:ns1="urn:acco
unting_2017_1.lists.webservices.netsuite.com">
               <ns1:matrixType xsi:type="ns2:ItemMatrixType" xmlns:ns2="urn:types.accounting_20
17_1.lists.webservices.netsuite.com">_parent</ns1:matrixType>
               <ns1:itemId xsi:type="xsd:string">sweater</ns1:itemId>
            </record>
            <record externalId="sweater-Red-Large" xsi:type="ns3:InventoryItem" xmlns:ns3="urn:
accounting_2017_1.lists.webservices.netsuite.com">
               <ns3:matrixType xsi:type="ns4:ItemMatrixType" xmlns:ns4="urn:types.accounting_20
17_1.lists.webservices.netsuite.com">_child</ns3:matrixType>
               <ns3:itemId xsi:type="xsd:string">sweater-Red-Large</ns3:itemId>
               <ns3:parent externalId="parentSweater" xsi:type="ns5:RecordRef" xmlns:ns5="urn:c
ore_2017_1.platform.webservices.netsuite.com"/>
               <ns3:matrixOptionList xsi:type="ns3:MatrixOptionList">
                  <ns3:matrixOption scriptId="CUSTITEM_COLOR" xsi:type="ns6:SelectCustomFieldRe
f" xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.com">
                     <ns6:value internalId="1" typeId="1" xsi:type="ns6:ListOrRecordRef">

                        <ns6:name xsi:type="xsd:string">Red</ns6:name>
                     </ns6:value>
                  </ns3:matrixOption>
                  <ns3:matrixOption scriptId="CUSTITEM_SIZE" xsi:type="ns7:SelectCustomFieldRef
" xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">
                     <ns7:value internalId="2" typeId="2" xsi:type="ns7:ListOrRecordRef">

                        <ns7:name xsi:type="xsd:string">Large</ns7:name>
                     </ns7:value>
                  </ns3:matrixOption>
               </ns3:matrixOptionList>
            </record>
            <record externalId="sweater-Green-Small" xsi:type="ns8:InventoryItem" xmlns:ns8="ur
n:accounting_2017_1.lists.webservices.netsuite.com">
               <ns8:matrixType xsi:type="ns9:ItemMatrixType" xmlns:ns9="urn:types.accounting_20
17_1.lists.webservices.netsuite.com">_child</ns8:matrixType>
               <ns8:itemId xsi:type="xsd:string">sweater-Green-Small</ns8:itemId>
               <ns8:parent externalId="parentSweater" xsi:type="ns10:RecordRef" xmlns:ns10="urn
:core_2017_1.platform.webservices.netsuite.com"/>
               <ns8:matrixOptionList xsi:type="ns8:MatrixOptionList">
                  <ns8:matrixOption scriptId="CUSTITEM_COLOR" xsi:type="ns11:SelectCustomFieldR
ef" xmlns:ns11="urn:core_2017_1.platform.webservices.netsuite.com">
                     <ns11:value internalId="2" typeId="1" xsi:type="ns11:ListOrRecordRef">

                        <ns11:name xsi:type="xsd:string">Green</ns11:name>
                     </ns11:value>
                  </ns8:matrixOption>
```

```
                   <ns8:matrixOption scriptId="CUSTITEM_SIZE" xsi:type="ns12:SelectCustomFieldRe
f" xmlns:ns12="urn:core_2017_1.platform.webservices.netsuite.com">
                      <ns12:value internalId="3" typeId="2" xsi:type="ns12:ListOrRecordRef">

                         <ns12:name xsi:type="xsd:string">Small</ns12:name>
                      </ns12:value>
                   </ns8:matrixOption>
                </ns8:matrixOptionList>
             </record>
             <record externalId="sweater-Blue-Large" xsi:type="ns13:InventoryItem" xmlns:ns13="u
rn:accounting_2017_1.lists.webservices.netsuite.com">
                <ns13:matrixType xsi:type="ns14:ItemMatrixType" xmlns:ns14="urn:types.accounting
_2017_1.lists.webservices.netsuite.com">_child</ns13:matrixType>
                <ns13:itemId xsi:type="xsd:string">sweater-Blue-Large</ns13:itemId>
                <ns13:parent externalId="parentSweater" xsi:type="ns15:RecordRef" xmlns:ns15="ur
n:core_2017_1.platform.webservices.netsuite.com"/>
                <ns13:matrixOptionList xsi:type="ns13:MatrixOptionList">
                   <ns13:matrixOption scriptId="CUSTITEM_COLOR" xsi:type="ns16:SelectCustomField
Ref" xmlns:ns16="urn:core_2017_1.platform.webservices.netsuite.com">
                      <ns16:value internalId="3" typeId="1" xsi:type="ns16:ListOrRecordRef">

                         <ns16:name xsi:type="xsd:string">Blue</ns16:name>
                      </ns16:value>
                   </ns13:matrixOption>
                   <ns13:matrixOption scriptId="CUSTITEM_SIZE" xsi:type="ns17:SelectCustomFieldR
ef" xmlns:ns17="urn:core_2017_1.platform.webservices.netsuite.com">
                      <ns17:value internalId="2" typeId="2" xsi:type="ns17:ListOrRecordRef">

                         <ns17:name xsi:type="xsd:string">Large</ns17:name>
                      </ns17:value>
                   </ns13:matrixOption>
                </ns13:matrixOptionList>
             </record>
             <record externalId="sweater-Red-Small" xsi:type="ns18:InventoryItem" xmlns:ns18="ur
n:accounting_2017_1.lists.webservices.netsuite.com">
                <ns18:matrixType xsi:type="ns19:ItemMatrixType" xmlns:ns19="urn:types.accounting
_2017_1.lists.webservices.netsuite.com">_child</ns18:matrixType>
                <ns18:itemId xsi:type="xsd:string">sweater-Red-Small</ns18:itemId>
                <ns18:parent externalId="parentSweater" xsi:type="ns20:RecordRef" xmlns:ns20="ur
n:core_2017_1.platform.webservices.netsuite.com"/>
                <ns18:matrixOptionList xsi:type="ns18:MatrixOptionList">
                   <ns18:matrixOption scriptId="CUSTITEM_COLOR" xsi:type="ns21:SelectCustomField
Ref" xmlns:ns21="urn:core_2017_1.platform.webservices.netsuite.com">
                      <ns21:value internalId="1" typeId="1" xsi:type="ns21:ListOrRecordRef">

                         <ns21:name xsi:type="xsd:string">Red</ns21:name>
                      </ns21:value>
                   </ns18:matrixOption>
                   <ns18:matrixOption scriptId="CUSTITEM_SIZE" xsi:type="ns22:SelectCustomFieldR
ef" xmlns:ns22="urn:core_2017_1.platform.webservices.netsuite.com">
                      <ns22:value internalId="3" typeId="2" xsi:type="ns22:ListOrRecordRef">

                         <ns22:name xsi:type="xsd:string">Small</ns22:name>
                      </ns22:value>
                   </ns18:matrixOption>
```

```
               </ns18:matrixOptionList>
            </record>
            <record externalId="sweater-Green-Large" xsi:type="ns23:InventoryItem" xmlns:ns23="
urn:accounting_2017_1.lists.webservices.netsuite.com">
               <ns23:matrixType xsi:type="ns24:ItemMatrixType" xmlns:ns24="urn:types.accounting
_2017_1.lists.webservices.netsuite.com">_child</ns23:matrixType>
               <ns23:itemId xsi:type="xsd:string">sweater-Green-Large</ns23:itemId>
               <ns23:parent externalId="parentSweater" xsi:type="ns25:RecordRef" xmlns:ns25="ur
n:core_2017_1.platform.webservices.netsuite.com"/>
               <ns23:matrixOptionList xsi:type="ns23:MatrixOptionList">
                  <ns23:matrixOption scriptId="CUSTITEM_COLOR" xsi:type="ns26:SelectCustomField
Ref" xmlns:ns26="urn:core_2017_1.platform.webservices.netsuite.com">
                     <ns26:value internalId="2" typeId="1" xsi:type="ns26:ListOrRecordRef">

                        <ns26:name xsi:type="xsd:string">Green</ns26:name>
                     </ns26:value>
                  </ns23:matrixOption>
                  <ns23:matrixOption scriptId="CUSTITEM_SIZE" xsi:type="ns27:SelectCustomFieldR
ef" xmlns:ns27="urn:core_2017_1.platform.webservices.netsuite.com">
                     <ns27:value internalId="2" typeId="2" xsi:type="ns27:ListOrRecordRef">

                        <ns27:name xsi:type="xsd:string">Large</ns27:name>
                     </ns27:value>
                  </ns23:matrixOption>
               </ns23:matrixOptionList>
            </record>
            <record externalId="sweater-Blue-Small" xsi:type="ns28:InventoryItem" xmlns:ns28="u
rn:accounting_2017_1.lists.webservices.netsuite.com">
               <ns28:matrixType xsi:type="ns29:ItemMatrixType" xmlns:ns29="urn:types.accounting
_2017_1.lists.webservices.netsuite.com">_child</ns28:matrixType>
               <ns28:itemId xsi:type="xsd:string">sweater-Blue-Small</ns28:itemId>
               <ns28:parent externalId="parentSweater" xsi:type="ns30:RecordRef" xmlns:ns30="ur
n:core_2017_1.platform.webservices.netsuite.com"/>
               <ns28:matrixOptionList xsi:type="ns28:MatrixOptionList">
                  <ns28:matrixOption scriptId="CUSTITEM_COLOR" xsi:type="ns31:SelectCustomField
Ref" xmlns:ns31="urn:core_2017_1.platform.webservices.netsuite.com">
                     <ns31:value internalId="3" typeId="1" xsi:type="ns31:ListOrRecordRef">

                        <ns31:name xsi:type="xsd:string">Blue</ns31:name>
                     </ns31:value>
                  </ns28:matrixOption>
                  <ns28:matrixOption scriptId="CUSTITEM_SIZE" xsi:type="ns32:SelectCustomFieldR
ef" xmlns:ns32="urn:core_2017_1.platform.webservices.netsuite.com">
                     <ns32:value internalId="3" typeId="2" xsi:type="ns32:ListOrRecordRef">
                        <ns32:name xsi:type="xsd:string">Small</ns32:name>
                     </ns32:value>
                  </ns28:matrixOption>
               </ns28:matrixOptionList>
            </record>
         </addList>
      </soapenv:Body>
   </soapenv:Envelope>
```

Java

```
/** Create Sweaters as matrix items.
* First create the parent - no matrix properties except "Matrix Type" is Parent
* Second create the matrix children with a combination of sizes and colors.
* This can be done in a single addList (as shown).
*/
//Define mrr method
public static RecordRef mrr(String internalId)
 {
  RecordRef toRet = new RecordRef();
  toRet.setInternalId(internalId);
  return toRet;
 }

// Define makeListOrRecordRef method
public static ListOrRecordRef makeListOrRecordRef(String sTypeId, String internalId, String sNa
me)
 {
  ListOrRecordRef toRet = new ListOrRecordRef();
  toRet.setInternalId(internalId);
  toRet.setName(sName);
  toRet.setTypeId(sTypeId);
  return toRet;
 }

public void testMatrixSample() throws Exception
   {
// Color is a Custom List of TypeId/RecType 1 that has already been created. 1,2,3 represent th
e
// internalIds of Red, Green, Blue
ListOrRecordRef[] colorArray = new
   ListOrRecordRef[] {makeListOrRecordRef("1","1","Red"), makeListOrRecordRef("1","2","Green"),

   makeListOrRecordRef("1","3","Blue")}; // Representing red, green and blue
// Size is a CustomList of TypeId/RecType 2 that has already been created
ListOrRecordRef[] sizeArray = new ListOrRecordRef[]{makeListOrRecordRef("2","2","Large"),makeLi
stOrRecordRef("2","3","Small")};


//Representing large and small
      InventoryItem[] toSubmit = new InventoryItem[1+colorArray.length*sizeArray.length];
      toSubmit[0] = new InventoryItem();
      toSubmit[0].setExternalId("parentSweater");
      toSubmit[0].setItemId("sweater");
      toSubmit[0].setMatrixType(ItemMatrixType._parent);
      // set other fields on the Parent

      for (int i=0;i<colorArray.length*sizeArray.length;i++)
      {
         toSubmit[i+1] = new InventoryItem();
         toSubmit[i+1].setMatrixType(ItemMatrixType._child);
         // mrr Creates a recordRef given an internal and externalId, the latter of which we sp
ecify.
         // This makes it so we can submit all the records at one time
         toSubmit[i+1].setParent(mrr((String)null,"parentSweater"));
         // "sweater-large-red","sweater-large-green"...
         toSubmit[i+1].setItemId("sweater-"+colorArray[i%3].getName() + "-" +
            sizeArray[i % 2].getName());
         // set externalId so it's easier to find later
         toSubmit[i+1].setExternalId(toSubmit[i+1].getItemId());
         // CUSTITEM_COLOR,SIZE are the names of the Item Custom Fields, applied to
         //InventoryItem that were setup as a Matrix types.
         SelectCustomFieldRef colorRef = new SelectCustomFieldRef();
         colorRef.setScriptId("CUSTITEM_COLOR");
         colorRef.setValue(colorArray[i%3]);
         SelectCustomFieldRef sizeRef = new SelectCustomFieldRef();
         sizeRef.setScriptId("CUSTITEM_SIZE");
         sizeRef.setValue(sizeArray[i%2]);
         toSubmit[i+1].setMatrixOptionList(new MatrixOptionList(new
            SelectCustomFieldRef[]{colorRef,sizeRef}));


         // Set other matrix item child files
         //....
      }

      WriteResponseList wr = c.getPort().addList(toSubmit);
   }
```

Working with the Pricing Matrix

The following method takes an InventoryItem record and sets its pricing matrix. It sets the base price at quantity 0. If an invalid price was entered (a non-numeric value), it does not set the pricing matrix.

Sample Java Code

```
private void createPricingMatrix(InventoryItem item)
{
   _console.write("\nPlease enter the base price: ");
   String priceString = _console.readLn();
   Price[] prices = new Price[1];
   prices[0] = new Price();
   try
   {
   prices[0].setValue(Double.valueOf(priceString));
   prices[0].setQuantity(null);

   PriceList priceList = new PriceList();
   priceList.setPrice(prices);

   Pricing[] pricing = new Pricing[1];
   pricing[0] = new Pricing();
   pricing[0].setPriceList(priceList);
   pricing[0].setDiscount(null);
   RecordRef priceLevel = new RecordRef();
   priceLevel.setInternalId(BASE_PRICE_LEVEL_INTERNAL_ID);
   priceLevel.setType(RecordType.priceLevel);
```

```
    pricing[0].setPriceLevel(priceLevel);
    RecordRef currUSD = new RecordRef();
    currUSD.setInternalId("1");
    pricing[0].setCurrency(currUSD);

    PricingMatrix pricingMatrix = new PricingMatrix();
    pricingMatrix.setPricing(pricing);
    pricingMatrix.setReplaceAll(false);

    item.setPricingMatrix(pricingMatrix);
    }
    catch (NumberFormatException e)
    {
       _console.error("\nInvalid base price entered: " + priceString + ".
          Proceed creating item without setting pricing matrix.");
    }
}
```

Shared Field Definitions for Items

The tables in this section provide field definitions for complex types that are shared across multiple item types. See the following shared item lists:

- Pricing Matrix List
- Billing Rates Matrix List
- Item Member List
- Item Options List
- Translation List
- Item Vendor List
- Site Category List

Pricing Matrix List

The Pricing Matrix List provides various pricings for a specific item. This list is supported when one or more of the following features are enabled: Multiple Currencies, Multiple Prices, and/or Quantity Pricing. For each item, you can set multiple pricings based on the following:

- Currency the item is offered in
- Price levels available for the item — as defined in the price level user defined list
- The quantity of items being sold — when the Quantity Pricing feature is enabled

Field Name	Type / Field Length	Req.	Mapping	Notes
currency	RecordRef	Y/N	NA	References the currency the price level will be set for. This field is required when the multi-currency preference is ON. Otherwise, the default currency is sourced for this field and it is NOT required.

Field Name	Type / Field Length	Req.	Mapping	Notes
discount	double	Y	Pricing / Default Discount %	This is a read-only field that returns the discount rate associated with the priceLevel.
priceLevel	RecordRef	Y	Pricing / Price Level	This is a read-only field that references values in a user-definable list at Setup > Accounting > Setup Tasks > Accounting Lists > New > Price Level. For details on how to edit this list, see Price List.
pricelist	List	N		See Price List.

Price List

To provide multiple entries in the Price List, the Quantity Pricing feature must be enabled. Otherwise, only one Price List entry should be submitted. Notice that in the UI, the quantity for multiple pricings is entered only one time. In web services, however, you must submit the quantity for **each** pricing. The quantity values for each price submitted in the same list **must match**. If they do NOT match, an error is returned.

Field Name	Type / Field Length	Req.	Mapping	Notes
quantity	double		Qty	Sets the quantity required to receive this price.
value	double		Pricing / amount text box	This is calculated based on the Base Price level and discount rate for the specified priceLevel but can be overwritten.

Pricing Matrix Keys

The key for the Pricing Matrix List varies according to the related features that are enabled. The key is the field or combination of fields with values that are used to uniquely identify each line in the list. The following table describes possible key combinations.

Enabled Feature(s)	Key Field(s)
Multiple Currencies only	currency
Multiple Currencies, Multiple Prices	combination of currency, priceLevel
Multiple Currencies, Quantity Pricing	combination of currency, quantity
Multiple Currencies, Multiple Prices, Quantity Pricing	combination of Currency,priceLevel, quantity
Multiple Prices only	priceLevel
Multiple Prices, Quantity Pricing	combination of priceLevel, quantity
Quantity Pricing Only	quantity

Important: In endpoints prior to 2012.1, pricingMatrixList updates ignore the replaceAll attribute and all requests behave as if it is set to TRUE. For the 2012.1 and later endpoints, replaceAll is respected. If you upgrade to one of these later endpoints, you need to carefully evaluate which replaceAll setting to use for pricing updates. For details about how replaceAll works with matrix sublists, see the help topic Matrix Sublists and replaceAll.

Billing Rates Matrix List

Use the Billing Rates list to set pricing for each billing class on service item records.

Because the Billing Classes feature is not compatible with the Quantity Pricing feature, you cannot use both features at the same time. If both are enabled, then billing classes replace quantity pricing on service item records.

Billing Rates

Field Name	Type / Field Length	Req.	Mapping	Notes
billingClass	RecordRef		Billing Rates / Billing Class	References a value in a user defined list at Setup > Accounting > Setup Tasks > Accounting Lists > New > Billing Class.
currency	RecordRef			References a value in a user-defined list at Lists > Accounting > Currencies. (Note that the Multiple Currencies feature must be enabled before you can set currency values.) This value sets the currency that all transactions involving this customer are conducted in. If defaults are OFF, this field is required. To retrieve a list of available values for this field, use the GetSelectValue operation. For more information, see the help topic getSelectValue of the Platform Guide.
rateList	List		Billing Rates	See Rate.

Rate

Field Name	Type / Field Length	Req.	Mapping	Notes
priceLevel	RecordRef		Billing Rates / [Various Price Level]	Sets the price Level this value is for. Price Levels are defined in a user defined list at Setup

Field Name	Type / Field Length	Req.	Mapping	Notes
				> Accounting > Setup Tasks > Accounting Lists > New > Price Level. For details on how to edit this list via web services, see Price Level.
value	double		Billing Rates / Amount field for each price level	Sets the amount for the associated priceLevel.

Item Member List

Field Name	Type / Field Length	Req.	Mapping	Notes
item	RecordRef			
memberDescr	string			
memberUnit	string			
quantity	double			
replaceAll	boolean			
taxcode	string			
taxrate	double			
taxSchedule	RecordRef			

Item Options List

Field Name	Type / Field Length	Req.	Mapping	Notes
itemOptions	recordRef	N	Custom / Item Options	References an existing transaction item options record at Setup > Customization > Transaction Item Options.

Translation List

Field Name	Type / Field Length	Req.	Mapping	Notes
description	string			
displayName	string			

Field Name	Type / Field Length	Req.	Mapping	Notes
featuredDescription	string		Featured Description	
language	string			
name	string			
noPriceMessage	string			
outOfStockMessage	string			
pageTitle	string	N	Store / Page Title	Sets the display title in the upper-left corner of an Internet browser when customers view this item in your Web store.
replaceAll	boolean			
salesDescription	string	N	Basic / Sales Description	Sets the description displayed when an item's store display name is clicked.
specialsDescription	string / 4000	N	Specials / Specials Description	Settable only if onSpecial is set to True. You can provide letters, numbers and basic HTML code.
storeDescription	string / 999	N	Store / Store Description	Sets the item description. This field can contain plain text as well as basic html code.
storeDetailedDescription	string / 4000	N	Store / Detailed Description	Sets the detailed item description. This field can contain plain text as well as basic html code.
storeDisplayName	string			

Item Vendor List

Use the Vendors List to add a list of vendors that can be used for this item. This information can then be accessed from the item record and from the corresponding vendor record.

The Multiple Vendors feature must be enabled to define item codes and purchase prices for multiple vendors on each item record. To enable this feature, go to Setup > Company > Enable Features, on the Items & Inventory tab, in the Items section, check the Multiple Vendors box.

Field Name	Type / Field Length	Req.	Mapping	Notes
preferredVendor	boolean	N	Basic / Preferred	You can only set one vendor in the list as preferred. If this is set

Field Name	Type / Field Length	Req.	Mapping	Notes
				true for multiple vendors, then the last vendor in the list is set as true and all others revert to false.
purchasePrice	double	N	Basic / Purchase Price	Sets the purchase price for this item when purchased from this vendor.
schedule	RecordRef			This field points to the quantity pricing schedule record, but is unique to the vendor and can only be created from the vendor record. In the UI go to Financial > Schedules > New Pricing Schedule.
subsidiary	string	N	Subsidiary	
vendor	RecordRef	Y	Basic / Vendor	References an existing vendor record. This is required for each vendor being defined. To retrieve a list of available values for this field, use the GetSelectValue operation. For more information, see the help topic getSelectValue of the Platform Guide.
vendorCode	string / 15	N	Basic / Vendor	Sets the vendor's item code.
vendorCurrencyName	string			

Site Category List

This list defines the website category for the item.

Field Name	Type / Field Length	Req.	Mapping	Notes
category	RecordRef	Y	Site Category	References a value in a user-defined list at Lists > Web Site > Categories.
categoryDescription	string / 50	Y	Description	A read-only field that returns the description defined in the category record.
website	RecordRef			

Assembly Item (BOM Item)

Assembly item records are also referred to as bill of material items (BOM items).

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's assembly item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Accessing Inventory Detail Data

If the Advanced Bin / Numbered Inventory Management feature is enabled, assembly/BOM items that have Use Bins set to True include data from the Inventory Detail subrecord. This subrecord includes quantity on hand and quantity available values per bin number.

If this feature is enabled, you must use the 2011.2 endpoint or later to access the newly supported subrecord and the most up to date bin fields for assembly/BOM items. If you are upgrading from a pre-2011.2 endpoint, you need to update any web services code that accesses these fields.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Description Item

Description item records are used to create a description line you can add to your transactions. Use the description item to enter long descriptions, without amounts, as line items.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's description item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Discount Item

Discount item records are used to create discounts you can apply to your transactions. For more details, see the help topic Discount Items.

The discount item record is defined in the accounting XSD.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's discount item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Download Item

Create download item records for files you want customers to purchase and download in your Web store. Customers are charged per download item as opposed to per item. For example, if you want to charge customers for music downloads per song, you would create a download item record for each song. If you want to charge customers per album, you would create one download item record and attach each song on the album to the record.

In web services, the download item record is defined in the listAcct (accounting) XSD.

Important: The download item record type is not accessible until the **Sell Download Files** feature is enabled in your account. In web services an error is thrown if you attempt create, update, delete or search for a Download Item without first enabling the **Sell Download Files** feature. To enable this feature, a NetSuite administrator can go to Setup > Company > Enable Features > select the Items & Inventory subtab > select the Sell Downloadable Files check box > click Save.

For general information on the download item record, see these topics in the NetSuite Help Center:

- Inventory Management Overview
- Download Items
- Setting Up Items for the Web Site

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's download item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Please note the following when working with the download items record:

- The download items record does not have its own search interface. Like all other item types in NetSuite, you must use the Item Search record to retrieve information for this record type.
- Working with the download item is generally a two-step process. You must first create the download item record using the **add** operation, then you must use the **attach** operation to attach all associated files. Note that an error is thrown if you attempt to attach a file that does not already exist in NetSuite. If you choose, you can later use the **detach** operation to remove a file from a download item record.

Note: You can create both the download item record and file record in one step using the addList operation. However, to attach any file to the download item record, you will still need to call the attach operation.

- In web services there is a 10 MB file size limit for file upload. If you attempt to upload a file larger than 10 MB to attach to a download item record, an error is thrown.

- When you do a **get** on download item records, the details of the files attached to the record are not returned in your SOAP response. If you need to get the list of files attached to a download item, you must do an advanced search on the item record and specify the fields on the file record that you want returned. If you want to update the files attached to a download item record, you must update the file record directly.
- To search on the files associated with a download item record, use the file join on the Item Search record.

Sample Code

C#

```
NetSuiteService nss = new NetSuiteService();

       ItemSearchAdvanced itemSearchAdv = new ItemSearchAdvanced();

       // Create search criteria
       ItemSearch itemSearch = new ItemSearch();
       ItemSearchBasic itemSearchBasic = new ItemSearchBasic();

       // Set item search type to downloadItem
       SearchEnumMultiSelectField itemTypeFld = new SearchEnumMultiSelectField();
       String [] itemTypes = new String[1];
       itemTypes[0] = "_downloadItem";
       itemTypeFld.searchValue = itemTypes;
       itemTypeFld.operatorSpecified = true;
       itemTypeFld.@operator = SearchEnumMultiSelectFieldOperator.anyOf;
       itemSearchBasic.type = itemTypeFld;

       // Select search return columns
       ItemSearchRow itemRow = new ItemSearchRow();

       // Select to return file internal id
       FileSearchRowBasic fileRowBasic = new FileSearchRowBasic();

       SearchColumnSelectField [] selectColumns = new SearchColumnSelectField[1];
       SearchColumnSelectField selectColumn = new SearchColumnSelectField();
       selectColumns[0] = selectColumn;
       fileRowBasic.internalId = selectColumns;

       // Select to return file name
       SearchColumnStringField [] stringColumns = new SearchColumnStringField[1];
       SearchColumnStringField stringColumn = new SearchColumnStringField();
       stringColumns[0] = stringColumn;
       fileRowBasic.name = stringColumns;

       // Set file join
       itemRow.fileJoin = fileRowBasic;

       // Select to return item internalId
       ItemSearchRowBasic itemRowBasic = new ItemSearchRowBasic();
       itemRowBasic.internalId = selectColumns; // column definition can be reused
       itemRow.basic = itemRowBasic;
```

```
        // Set item search criteria
        itemSearch.basic = itemSearchBasic;
        itemSearchAdv.criteria = itemSearch;

        // Set item return columns
        itemSearchAdv.columns = itemRow;

        // Perform search
        SearchResult searchResult = nss.search(itemSearchAdv);
```

SOAP Request

```
        <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
           <searchRecord xmlns:q1="urn:accounting_2017_1.lists.webservices.netsuite.com" xsi:t
ype="q1:ItemSearchAdvanced">
              <q1:criteria>
                 <q1:basic>
                    <type operator="anyOf" xmlns="urn:common_2017_1.platform.webservices.netsu
ite.com">
                       <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
_downloadItem</searchValue>
                    </type>
                 </q1:basic>
              </q1:criteria>
              <q1:columns>
                 <q1:basic>
                    <internalId xmlns="urn:common_2017_1.platform.webservices.netsuite.com" />

                 </q1:basic>
                 <q1:fileJoin>
                    <internalId xmlns="urn:common_2017_1.platform.webservices.netsuite.com" />

                    <name xmlns="urn:common_2017_1.platform.webservices.netsuite.com" />
                 </q1:fileJoin>
              </q1:columns>
           </searchRecord>
        </search>
```

SOAP Response

```
              <platformCore:searchRowList>
                 <platformCore:searchRow xsi:type="listAcct:ItemSearchRow" xmlns:listAcct="urn
:accounting_2017_1.lists.webservices.netsuite.com">
                    <listAcct:basic xmlns:platformCommon="urn:common_2017_1.platform.webservic
es.netsuite.com">
                       <platformCommon:internalId>
                          <platformCore:searchValue internalId="105"/>
                       </platformCommon:internalId>
                    </listAcct:basic>
                    <listAcct:fileJoin xmlns:platformCommon="urn:common_2017_1.platform.webser
vices.netsuite.com">
                       <platformCommon:internalId>
                          <platformCore:searchValue internalId="239"/>
                       </platformCommon:internalId>
                       <platformCommon:name>
                          <platformCore:searchValue>IDreamedADreamLyrics.txt</platformCore:sea
rchValue>
                       </platformCommon:name>
                    </listAcct:fileJoin>
                 </platformCore:searchRow>
              </platformCore:searchRowList>
```

Gift Certificate Item

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's gift certificate item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

This sample shows how to add a gift certificate item. Note that the following has been set in the NetSuite account: Setup > Accounting > Accounting Preferences > Items/Transactions tab > Other Items Preferences > Gift Certificate Auth Code Generation = System Generated.

```
soapenv:Body>
      <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
         <record xsi:type="ns6:GiftCertificateItem" xmlns:ns6="urn:accounting_2017_1.lists.webs
ervices.netsuite.com">
            <ns6:itemId xsi:type="xsd:string">BeyondMotorSports.com Gift Certificate</ns6:itemI
d>
            <ns6:isInactive xsi:type="xsd:boolean">false</ns6:isInactive>
            <ns6:liabilityAccount internalId="28" type="account" xsi:type="ns7:RecordRef"
              xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com" />
            <ns6:rate xsi:type="xsd:double">100.0</ns6:rate>
            <ns6:taxSchedule internalId="1" xsi:type="ns8:RecordRef"
              xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com" />
         </record>
      </add>
```

```
   </soapenv:Body>
```

SOAP Response

```
  <soapenv:Body>
      <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
         <writeResponse>
            <ns2:status isSuccess="true" xmlns:ns2="urn:core_2017_1.platform.webservices.netsui
te.com" />
            <baseRef internalId="93" type="giftCertificateItem" xsi:type="ns3:RecordRef"
              xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com" />
         </writeResponse>
      </addResponse>
   </soapenv:Body>
```

Java

```
GiftCertificateItemAuthCodes authCode1 = new GiftCertificateItemAuthCodes();
authCode1.setAuthCode("AUTHCODE003");

GiftCertificateItemAuthCodes authCode2 = new GiftCertificateItemAuthCodes();
authCode2.setAuthCode("AUTHCODE004");

GiftCertificateItemAuthCodes[] authCodes = new GiftCertificateItemAuthCodes[2];
authCodes[0] = authCode1;
authCodes[1] = authCode2;

GiftCertificateItemAuthCodesList authCodesList = new GiftCertificateItemAuthCodesList();
authCodesList.setAuthCodes(authCodes);
authCodesList.setReplaceAll(true);

RecordRef liabilityAccount = new RecordRef();
liabilityAccount.setInternalId("28");
liabilityAccount.setType(RecordType.account);

RecordRef taxSchedule = new RecordRef();
taxSchedule.setInternalId("1");

GiftCertificateItem gcItem = new GiftCertificateItem();
gcItem.setItemId("BeyondMotorSports.com Gift Certificate");
gcItem.setLiabilityAccount(liabilityAccount);
gcItem.setTaxSchedule(taxSchedule);
gcItem.setRate(100.00);
gcItem.setIsInactive(false);

WriteResponse response = _port.add(gcItem);
WriteResponse response = _port.add(gci);
```

Inventory Item

Inventory item records are used to track information about items for which you maintain a stock.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s inventory item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Working with Inventory Items Sublists

The SuiteTalk Schema Browser includes all sublists associated with the inventory item record. See the following information for usage notes regarding specific Inventory Item sublists. Usage notes are not provided for every sublist type.

InventoryItemLocations

To provide a locations list, the Multi-Location Inventory feature must be enabled at Setup > Company > Enable Features. Click the Items & Inventory subtab, and select the Multi-Location Inventory check box. Otherwise, single entries for each corresponding field can be entered in the regular record fields.

ItemVendorList

The Vendor sublist (ItemVendorList) on inventory items contains a **schedule** field that holds pricing schedule values. Note that when this field is set, it triggers a recalc on add and update operations.

Getting Bin Details for Items

If the Bin Management feature is enabled, you can use ItemSearchAdvanced to get bin-related details about inventory items. The following code snippets illustrate a search that returns the quantity available per bin.

Accessing Inventory Detail Data

If the Advanced Bin / Numbered Inventory Management feature is enabled, inventory items that have Use Bins set to True include data from a new Inventory Detail subrecord. This subrecord includes quantity on hand and quantity available values per bin number.

If this feature is enabled, you must use the 2011.2 endpoint or later to access the newly supported subrecord and the most up to date bin fields for inventory items. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Examples

SOAP Request

```
<soapenv:Body>
         <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <searchRecord xsi:type="ns7:ItemSearchAdvanced" xmlns:ns7="urn:accounting_2017_1.li
sts.webservices.netsuite.com">
               <ns7:criteria xsi:type="ns7:ItemSearch">
...
              </ns7:criteria>
               <ns7:columns xsi:type="ns7:ItemSearchRow">
                  <ns7:basic xsi:type="ns10:ItemSearchRowBasic" xmlns:ns10="urn:common_2017_1.p
latform.webservices.netsuite.com">
                     <ns10:internalId xsi:type="ns11:SearchColumnSelectField" xmlns:ns11="urn:c
ore_2017_1.platform.webservices.netsuite.com"/>
                     <ns10:itemId xsi:type="ns12:SearchColumnStringField" xmlns:ns12="urn:core_
2017_1.platform.webservices.netsuite.com"/>
                     <ns10:locationBinQuantityAvailable xsi:type="ns13:SearchColumnStringField"
 xmlns:ns13="urn:core_2017_1.platform.webservices.netsuite.com"/>
                  </ns7:basic>
               </ns7:columns>
            </searchRecord>
         </search>
```

SOAP Response

```
<platformCommon:locationBinQuantityAvailable>
                           <platformCore:searchValue>AS1_BinUK(300),AS2_BinUK(200)</platformCor
e:searchValue>
                        </platformCommon:locationBinQuantityAvailable>
```

Note: As of the 2013.1 endpoint, locationBinQuantityAvailable has a type of SearchColumnStringField. In earlier endpoints, it had a type of SearchColumnDoubleField.

Item Group

The item group record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the item group record.

add | addList | delete | deleteList | getDeleted | get | getList | getSavedSearch | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's item group reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

The following sample shows how to add an item group through SuiteTalk. Note that the itemMemberList (Components) sublist is a non-keyed sublist.

Code Sample

Java

```
public void addItemGroup() throws Exception
   {
      this.login();

      ItemGroup ig = new ItemGroup();

      ig.setItemId("Test Item Group");
      ig.setDisplayName("Test IG");

      //Create a list of subsidiaries
      RecordRefList recordRefList = new RecordRefList();
      RecordRef rr = new RecordRef();
      rr.setInternalId("3");
      rr.setType(RecordType.subsidiary);
      recordRefList.setRecordRef(new RecordRef[]{rr});
      ig.setSubsidiaryList(recordRefList);

      RecordRef depRef = new RecordRef();
      depRef.setInternalId("2");
      ig.setDepartment(depRef);
      RecordRef locRef = new RecordRef();
      locRef.setInternalId("2");
      ig.setLocation(locRef);
      RecordRef classRef = new RecordRef();
      classRef.setInternalId("2");
      ig.set_class(classRef);
      //Item sublist
      ItemMemberList iml = new ItemMemberList();
      ItemMember im = new ItemMember();
      RecordRef itRef = new RecordRef();
      itRef.setInternalId("39");
      im.setItem(itRef);
      im.setQuantity(3.0);
      im.setVsoeDeferral(VsoeDeferral._deferUntilItemDelivered);
      im.setVsoeDelivered(true);
      im.setVsoePermitDiscount(VsoePermitDiscount._never);
      iml.setItemMember(new ItemMember[]{im});

      ig.setMemberList(iml);
      ig.setDescription("This is a test Item Group");
      ig.setIsVsoeBundle(false);
      ig.setAvailableToPartners(true);

      _port.add(ig);
   }
```

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>
        <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record xsi:type="ns6:ItemGroup" xmlns:ns6="urn:accounting_2017_1.lists.webservices
.netsuite.com">
                <ns6:isVsoeBundle xsi:type="xsd:boolean">false</ns6:isVsoeBundle>
                <ns6:availableToPartners xsi:type="xsd:boolean">true</ns6:availableToPartners>
                <ns6:itemId xsi:type="xsd:string">Test Item Group</ns6:itemId>
                <ns6:displayName xsi:type="xsd:string">Test IG</ns6:displayName>
                <ns6:description xsi:type="xsd:string">This is a test Item Group</ns6:descripti
on>
                <ns6:subsidiaryList xsi:type="ns7:RecordRefList" xmlns:ns7="urn:core_2017_1.pla
tform.webservices.netsuite.com">
                    <ns7:recordRef xsi:type="ns7:RecordRef" type="subsidiary" internalId="3"/>
                </ns6:subsidiaryList>
                <ns6:department xsi:type="ns8:RecordRef" internalId="2" xmlns:ns8="urn:core_201
7_1.platform.webservices.netsuite.com"/>
```

```
                <ns6:class xsi:type="ns9:RecordRef" internalId="2" xmlns:ns9="urn:core_2017_1.p
latform.webservices.netsuite.com"/>
                <ns6:location xsi:type="ns10:RecordRef" internalId="2" xmlns:ns10="urn:core_201
7_1.platform.webservices.netsuite.com"/>
                <ns6:memberList xsi:type="ns6:ItemMemberList" replaceAll="false">
                    <ns6:itemMember xsi:type="ns6:ItemMember">
                        <ns6:quantity xsi:type="xsd:double">3.0</ns6:quantity>
                        <ns6:vsoeDeferral xsi:type="ns11:VsoeDeferral" xmlns:ns11="urn:types.co
mmon_2017_1.platform.webservices.netsuite.com">_deferUntilItemDelivered</ns6:vsoeDeferral>
                        <ns6:vsoePermitDiscount xsi:type="ns12:VsoePermitDiscount" xmlns:ns12="
urn:types.common_2017_1.platform.webservices.netsuite.com">_never</ns6:vsoePermitDiscount>
                        <ns6:vsoeDelivered xsi:type="xsd:boolean">true</ns6:vsoeDelivered>
                        <ns6:item xsi:type="ns13:RecordRef" internalId="39" xmlns:ns13="urn:cor
e_2017_1.platform.webservices.netsuite.com"/>
                    </ns6:itemMember>
                </ns6:memberList>
            </record>
        </add>
    </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_1234567_081020122627225976106604 20_af57ceedff97e</pl
atformMsgs:nsId>
        </platformMsgs:documentInfo>
    </soapenv:Header>
    <soapenv:Body>
        <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <writeResponse>
                <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platf
orm.webservices.netsuite.com"/>
                <baseRef xsi:type="platformCore:RecordRef" type="itemGroup" internalId="204" xm
lns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </writeResponse>
        </addResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Kit/Package Item

Kits or packages let you create items that are collected from other items. You can assign multiple price levels to your kits and even make them available in your website. Whenever you sell a kit, inventory items are deducted from inventory.

For information on working with kits and packages in the user interface, see the help topic Kit/Package Items.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's kit item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Lot Numbered Assembly Item

Lot numbered assembly items enable you to build items from raw materials and track the inventory of both the finished items and the raw materials separately. The completed assembly is assigned a lot number to track it as it enters and leaves your inventory.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's lot numbered assembly item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Accessing Inventory Detail Data

If the Advanced Bin / Numbered Inventory Management feature is enabled, lot numbered assembly items include data from a new Inventory Detail subrecord. This subrecord includes quantity on hand and quantity available values per lot number, and if applicable, per bin number.

If this feature is enabled, you must use the 2011.2 endpoint or later to access the newly supported subrecord and the most up to date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Lot Numbered Inventory Item

Lot numbered inventory items track the purchase, stock, and sale of groups of items by assigning lot numbers. Lot numbered item records track the quantity of items and the specific cost for each lot as products are purchased and sold.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's lot numbered inventory item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Working with Lot Numbered Inventory Items Sublists

The SuiteTalk Schema Browser includes all sublists associated with the lot numbered inventory item record. See the following information for usage notes regarding specific lot numbered inventory item sublists. Usage notes are not provided for every sublist type.

LotNumberedInventoryItemLocations

To provide a locations list for lot numbered inventory Items, the Multi-Location Inventory feature must be enabled at Setup > Company > Enable Features > Accounting. Otherwise, single entries for each corresponding field can be entered in the regular record fields.

LotNumberedInventoryItemNumbers

This list is read-only and displays the serial number and quantity on hand for the lot numbered inventory item.

Accessing Inventory Detail Data

If the Advanced Bin / Numbered Inventory Management feature is enabled, lot numbered inventory items include data from a new Inventory Detail subrecord. This subrecord includes quantity on hand and quantity available values per lot number, and if applicable, per bin number.

If this feature is enabled, you must use the 2011.2 endpoint or later to access the newly supported subrecord and the most up to date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Markup Item

Markup item records are used to create markups you can apply to your transactions.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's markup item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Noninventory Purchase Item

Noninventory purchase item records are used to track something you buy but do not stock.

For more information, see the help topic Non-Inventory Items.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s noninventory purchase item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Noninventory Resale Item

Noninventory resale item records are used to track something you buy and then sell for a profit, but do not stock.

For more information, see the help topic Non-Inventory Items.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’ s noninventory resale item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

NonInventory Sales Item

Noninventory sales item records are used to track items you sell but do not stock.

For more information, see the help topic Non-Inventory Items.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's noninventory sales item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Other Charge Purchase Item

Other charge purchase item records are used to track charges that your business pays for that are not billed to a customer.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's other charge purchase item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Other Charge Resale Item

Other charge resale item records are used to track charges that your business pays for initially and then bills to a customer, like freight charges.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's other charge resale item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Other Charge Sale Item

Other charge sale item records are used to track charges for something a business creates or performs and charges a customer for, like a charge for gift wrapping.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's other charge sale item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Payment Item

Payment item records are used for partial payments, such as showing a down payment when you create an invoice. Payment items can be assigned to post to your un-deposited funds account.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's payment item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Serialized Assembly Item

Serialized assembly items enable you to build items from raw materials and track the inventory of both finished items and the raw materials separately. The completed assembly is assigned a serial number to track it as it enters and leaves your inventory. Serialized assembly items are available on sales transactions and inventory adjustment transactions. They are not available on purchase transactions.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's serialized assembly item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Accessing Inventory Detail Data

If the Advanced Bin / Numbered Inventory Management feature is enabled, serialized assembly items include data from the Inventory Detail subrecord. This subrecord includes quantity on hand and quantity available values per serial number, and if applicable, per bin number.

If this feature is enabled, you must use the 2011.2 endpoint or later to access the newly supported subrecord and the most up to date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Serialized Inventory Item

Serialized inventory item records are used to track information about items which you maintain a stock of. Note that you must first enable serialized inventory items in your NetSuite account before you can access this record type. To enable serialized inventory items, go to Setup > Company > Enable Features. On the Items & Inventory tab, under Inventory, select the Serialized Inventory check box.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's serialized inventory item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Working with Serialized Inventory Items Sublists

The SuiteTalk Schema Browser includes all sublists associated with the serialized inventory item record. See the following information for usage notes regarding specific serialized inventory item sublists. Usage notes are not provided for every sublist type.

SerializedInventoryItemLocations

To provide a locations list, the Multi-Location Inventory feature must be enabled at Setup > Company > Enable Features > Accounting. Otherwise, single entries for each corresponding field can be entered in the regular record fields.

SerializedInventoryItemNumbers

This list is read-only and displays the serial number and quantity on hand for the lot numbered inventory item.

Accessing Inventory Detail Data

If the Advanced Bin / Numbered Inventory Management feature is enabled, serialized inventory items include data from a new Inventory Detail subrecord. This subrecord includes quantity on hand and quantity available values per serial number, and if applicable, per bin number.

If this feature is enabled, you must use the 2011.2 endpoint or later to access the newly supported subrecord and the most up to date bin and numbered inventory fields. You need to update any web services code from a previous endpoint that accesses these fields, to avoid errors or unexpected results.

For more details, see Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Service Purchase Item

Service purchase item records are used to track charges for a service your business pays for that is not billed to a customer.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s service purchase item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Service Resale Item

Service resale item records are used to track charges for a service your business pays for initially and then bills to a customer.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s service resale item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Service Sale Item

Service sale item records are used to track charges for a service your business performs that is billed to a customer.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s service sale item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Subtotal Item

Subtotal item records are used to separate groups or individual line items if you want to offer a discount or tax on certain line items but not others.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's subtotal item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Support

The following support records are supported in SuiteTalk:

- Issue
- Support Case
- Support Case Status
- Support Case Type
- Support Case Origin
- Support Case Issue
- Support Case Priority
- Solution
- Topic

Issue

The issue record is defined in the listSupport (support) XSD.

For details about using issue records in the user interface, see the help topic Issue Management.

Supported Operations

The following operations can be used with Issue records.

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's issue reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Be aware that the behavior of the issue record varies depending on the value of the Use Multiple Versions and Builds preference, which you set at Setup > Issues > Issue Preferences. When you select the **Use Multiple Versions and Builds** box, the issue record uses three sublists: customFieldList, fixedInVersionList, and relatedIssuesList. When you do not set this preference, the record uses the following six body fields:

- versionBroken

- buildBroken
- versionTarget
- buildTarget
- versionFixed
- buildFixed

Note that these body fields are exposed in the 2014.1 and later WSDLs, but not in previous WSDLs.

Support Case

Support Cases are support issues logged for a specific company.

The SupportCase record is defined in the listSupport (support) XSD.

For details about using case records in the user interface, see the help topic Setting Up Case Management.

Supported Operations

The following operations can be used with the support case record.

add | addList | attach / detach | delete | deleteList | get | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Note: The getDeleted operation is NOT supported for Cases.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's support case reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Case Status and Priority Internal IDs

The following table lists the internal IDs for all standard Case Status and Priority values that can be used to populate the status and priority fields.

The status and priority values can be deleted or recreated so that the following internal ID values may differ for your organization. If the Show Internal IDs preference is enabled, you can confirm the internal ID values in the associated list. See the help topic Enabling the Show Internal IDs Preference.

Case Statuses		Case Priorities	
ID	**Status**	**ID**	**Priority**
1	Not Started	1	High
2	In Progress	2	Medium
3	Escalated	3	Low
4	Re-Opened		
5	Closed		

Working with Support Case Sublists

The SuiteTalk Schema Browser includes all sublists associated with the support case record. See the following information for usage notes regarding specific support case sublists. Usage notes are not provided for every sublist type.

EmailEmployeesList

Use this list to provide a list of employees that should be copied when the record is updated. Values submitted in this list are not saved in the NetSuite database.

MessageList

The MessageList complex type has been deprecated as of SuiteTalk Version 2_0_0.

Support Case Status

Case status records define available statuses for customer support cases. The SupportCaseStatus record is defined in the listSupport (support) XSD.

For details about support case statuses in the user interface, see the help topic Creating Case Statuses.

Supported Operations

The following operations can be used with the support case status record.

add | addList | delete | deleteList | get | getAll | getDeleted | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s support case status reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Support Case Type

Case type records define available case types for customer support cases. The SupportCaseType record is defined in the listSupport (support) XSD.

For details about support case types in the user interface, see the help topic Creating Case Types.

Supported Operations

The following operations can be used with the support case type record.

add | addList | delete | deleteList | get | getAll | getDeleted | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's support case type reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Support Case Origin

Case origin records define the various methods of origin for customer support cases. The SupportCaseOrigin record is defined in the listSupport (support) XSD.

For details about support case origins in the user interface, see the help topic Creating Case Origins.

Supported Operations

The following operations can be used to manipulate the support case origin record.

add | addList | delete | deleteList | get | getAll | getDeleted | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's support case origin reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Support Case Issue

Support Case Issue records define available issue types for customer support cases. The SupportCaseIssue record is defined in the listSupport (support) XSD.

For details about support case issues in the user interface, see the help topic Creating Case Issues.

Supported Operations

The following operations can be used with the support case issue record.

add | addList | delete | deleteList | get | getAll | getDeleted | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's support case reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Support Case Priority

Case priority records define available case priorities for customer support cases. The SupportCasePriority record is defined in the listSupport (support) XSD.

For details about support case priorities in the user interface, see the help topic Creating Case Priorities.

Supported Operations

The following operations can be used with the support case priority record.

add | addList | delete | deleteList | get | getAll | getDeleted | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's support case priority reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Solution

The solution record is defined in the listsSupport (support) XSD.

This record is available when the Knowledge Base feature is enabled at Setup > Company > Enable Features, on the CRM tab. When the feature is enabled, you can access the inventory cost revaluation record in the UI by choosing Lists > Support > Solutions > New.

For details about working with this record manually, see the help topic Creating Knowledge Base Solutions.

Supported Operations

The following operations can be used with the solution record.

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's solution reference page. For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Topic

The topic record is defined in the listsSupport (support) XSD.

For details about topics in the user interface, see the help topic Creating Knowledge Base Topics.

Supported Operations

The following operations can be used with the topic record.

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's topic reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Website

One website record is supported in SuiteTalk — Site Category.

Site Category

The site category record is defined in the listsWebsite (website) XSD.

For more information on the NetSuite PHP toolkit, see the help topic Web Services PHP Toolkit in the NetSuite Help Center.

Supported Operations

The following operations can be used with the site category record:

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s site category reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Root site categories (such as the Welcome Page) are not editable in web services because they have negative internalIds. For example, the following update operation will fail when submitted:

```
<update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <record internalId="-102" xsi:type="ns1:SiteCategory"
   xmlns:ns1="urn:website_2017_1.lists.webservices.netsuite.com">
    <ns1:pageTitle xsi:type="xsd:string">pageTitle</ns1:pageTitle>
  </record>
</update>
```

where:

internalId="-102" is the internal ID of a root category

If you attempt to edit/update root site categories, the following error message is returned:

Cannot update root level website categories through Web Services.

Lists

The following list records are supported in SuiteTalk:

- Account
- Accounting Period
- Address
- Billing Account
- Billing Schedule
- Bin
- Classification (Class)
- Consolidated Exchange Rate
- Currency
- Currency Rate
- Department
- Expense Category
- Fair Value Price
- Gift Certificate
- Inventory Number
- Item Revision
- Location
- Manufacturing Cost Template
- Manufacturing Routing
- Nexus
- Payroll Item
- Revenue Recognition Schedule
- Revenue Recognition Template
- Sales Tax Item
- Subsidiary
- Tax Group
- Tax Type
- Unit Type

Account

The account record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the account record.

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s account reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Note: The unit and unitstype fields are used only for statistical accounts, not other types of accounts. For general details on statistical accounts, see the help topic Statistical Accounting Overview.

Usage Notes

Currently there are no available search joins for this record.

Certain Fields are Available for OneWorld Accounts Only

The following fields are available for OneWorld accounts only:

- accountingContext. At least one Accounting Context must exist in your account for this field to be available.
- acctNumber. The Use Account Numbers accounting preference must be enabled for this field to be available.

Be aware of the following limitations for these fields in 2016.1 and earlier endpoints:

- get operations on the account record return only lines that have null values for the accountingContext and acctNumber fields.
- You cannot set values for the accountingContext and acctNumber fields in the localization sublist.

Get Response

The following snippet shows a SOAP response to a get request on the account record.

```
<listAcct:localizationsList>
   <listAcct:accountLocalizations>
      <listAcct:accountingContext internalId="2" xmlns:platformCore="urn:core_2017_1.platform.w
ebservices.netsuite.com">
         <platformCore:name>European Accounting Context</platformCore:name>
      </listAcct:accountingContext>
      <listAcct:acctName>France</listAcct:acctName>
```

```
        <listAcct:locale>_frenchFrance</listAcct:locale>
   </listAcct:accountLocalizations>
      <listAcct:accountLocalizations>
      <listAcct:acctName>Germany</listAcct:acctName>
      <listAcct:locale>_german</listAcct:locale>
   </listAcct:accountLocalizations>
</listAcct:localizationsList>
```

Java Update Request

The following Java snippet uses the acctContext field in an update request.

```
Account account = new Account();

account.setInternalId("161");
AccountLocalizationsList accountLocalizationsList = new AccountLocalizationsList();
AccountLocalizations[] accountLocalizations = {new AccountLocalizations()};
accountLocalizations[0].setLocale(Language._russian);
accountLocalizations[0].setAcctName("Office Expenses");
RecordRef acctContext = new RecordRef();
acctContext.setInternalId("1");
accountLocalizations[0].setAccountingContext(acctContext);
accountLocalizationsList.setAccountLocalizations(accountLocalizations);
account.setLocalizationsList(accountLocalizationsList);

c.updateRecord(account);
```

Update Request

The following SOAP snippet uses the acctContext field in an update request.

```
<update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com>
      <record internalId="161" xsi:type="ns8:Account" xmlns:ns8="urn:accounting_2017_1.lists.we
bservices.netsuite.com">
         <ns8:localizationsList replaceAll="false" xsi:type="ns8:AccountLocalizationsList">

            <ns8:accountLocalizations xsi:type="ns8:AccountLocalizations">
               <ns8:accountingContext internalId="1" xsi:type="ns9:RecordRef" xmlns:ns9="urn:co
re_2017_1.platform.webservices.netsuite.com"/>
                <ns8:acctName xsi:type="xsd:string">Office Expenses</ns8:acctName>
                <ns8:locale xsi:type="ns10:Language" xmlns:ns10="urn:types.common_2017_1.platfo
rm.webservices.netsuite.com">_russian</ns8:locale>
            </ns8:accountLocalizations>
         </ns8:localizationsList>
      </record>
</update>
```

Accounting Period

In NetSuite, transactions are posted in real time. If you choose to use accounting periods, you must enter a posting period on the transaction and select its posting period from a dropdown list on the transaction page. The posting period can be future or past, depending on the close status of the period.

To access the accounting periods record in the UI, go to Setup > Accounting > Manage Accounting Periods. For information on setting up accounting periods, see the help topic Setting Up Accounting Periods in the NetSuite Help Center.

In web services, the accounting period record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the accounting period record.

get | getList | search | searchMore | searchMoreWithId | searchNext

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's accounting period reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Important: This record is available as a read-only record. To make changes to this record, you must do so through the UI.

Also note that to use the accounting periods record, a NetSuite administrator must first enable this feature in your account.

To enable Accounting Periods:

1. Go to Setup > Company > Enable Features.
2. Click the **Accounting** subtab.
3. Check the **Accounting Periods** box.
4. Click **Save**.

Billing Account

A billing account is a record used to show all billing information for a customer or subcustomer. A billing account contains billing-specific information, including billing schedule, default payment terms, bill-to address, and currency.

The billing account record is defined in the relationships XSD, where it is represented as the BillingAccount complex type.

To enable Billing Accounts, you must first enable:

- Consolidate Projects on Sales Transactions (Setup > Accounting > Accounting Preferences > Items/ Transactions subtab > Sales & Pricing > Consolidate Projects on Sales Transactions).
- Charge Based Billing (Setup > Company > Enable Features > Transactions subtab> Billing > Charge Based Billing).

After enabling both prerequisites, enable Billing Accounts by going to Setup > Company > Enable Features > Transactions subtab > Billing > Billing Accounts.

Access Billing accounts from Lists > Relationships > Billing Accounts or by clicking New on the Billing Account subtab on the Customer record.

Note: If you have multiple billing accounts for a customer, the billing account marked as customer default is used when creating transactions, for example a new invoice.

For more on using web services to interact with billing account, see the following topics:

- Billing Account Supported Operations
- Billing Account Body Fields
- Billing Account Code Samples

Billing Account Supported Operations

add | addList | delete | deleteList | get | | getList | search | searchMore | searchNext | searchMoreWithId | getSavedSearch |update | updateList | upsert | upsertList |

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Billing Account Body Fields

When creating a usage record, you are required to define the following body fields:

Field Name	Label in UI	Description/Type
`customer`	Customer	The customer associated with the billing account. This is a RecordRef field.
`currency`	Currency	The currency associated with the billing account. This is a RecordRef field.
`billingSchedule`	Billing Schedule	The billing account schedule. This is a RecordRef field.
`startDate`	Start Date	The date on which the billing account starts. This is a dateTime field. You can only change the start date of a billing account if it is not yet tied to a subscription.

The SuiteTalk Schema Browser includes definitions for all body fields, search filters, and search joins available to this record. For details, see the Schema Browser's billing account reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Billing Account Code Samples

For examples of how to work with billing account records using SuiteTalk, refer to the following sections:

- Adding a Standard Billing Account
- Adding a Billing Account With Preferences

Adding a Standard Billing Account

This example shows how to add a standard billing account record.

Java

```
      //Create the record

      BillingAccount record = new BillingAccount();

      //Set fields on the billing account record
      record.setName("Benbow Billing Account");
      record.setCustomer(createRR("305"));
      record.set_class(createRR("1"));
      record.setBillingSchedule(createRR("2"));
      record.setFrequency(BillingAccountFrequency._monthly)

      //Set Start Date field using the Gregorian Calendar
      Calendar startDate = new GregorianCalendar();
      startDate.set(Calendar.YEAR, 2017);
      startDate.set(Calendar.MONTH, 0);
      startDate.set(Calendar.DAY_OF_MONTH, 15);
      record.setStartDate(startDate);

      //Add the record
      c.addRecord(record);
```

SOAP Request

```
<soapenv:Body>
        <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record xsi:type="ns8:BillingAccount" xmlns:ns8="urn:relationships_2017_1.lists.web
services.netsuite.com">
                <ns8:customer xsi:type="ns9:RecordRef" internalId="305" xmlns:ns9="urn:core_201
7_1.platform.webservices.netsuite.com"/>
                <ns8:name xsi:type="xsd:string">Benbow Billing Account</ns8:name>
                <ns8:class xsi:type="ns10:RecordRef" internalId="1" xmlns:ns10="urn:core_2017_1
.platform.webservices.netsuite.com"/>
```

```
                <ns8:billingSchedule xsi:type="ns11:RecordRef" internalId="2" xmlns:ns11="urn:c
ore_2017_1.platform.webservices.netsuite.com"/>
                <ns8:frequency xsi:type="ns12:BillingAccountFrequency" xmlns:ns12="urn:types.re
lationships_2017_1.lists.webservices.netsuite.com">_monthly</ns8:frequency>
                <ns8:startDate xsi:type="xsd:dateTime">2017-01-15T14:37:49.519Z</ns8:startDate>

            </record>
        </add>
    </soapenv:Body>
```

Adding a Billing Account With Preferences

The following example adds a billing account record that specifies the invoice form.

Java

```
      //Create the record

      BillingAccount record = new BillingAccount();

      //Set fields on the billing account record
      record.setName("FedEx Billing Account");
      record.setCustomer(createRR("305"));
      record.set_class(createRR("1"));
      record.setBillingSchedule(createRR("2"));
      record.setFrequency(BillingAccountFrequency._monthly)
      //set the Invoice Form field
      record.setInvoiceForm(createRR("69");

      //Set Start Date field using the Gregorian Calendar
      Calendar startDate = new GregorianCalendar();
      startDate.set(Calendar.YEAR, 2017);
      startDate.set(Calendar.MONTH, 0);
      startDate.set(Calendar.DAY_OF_MONTH, 15);
      record.setStartDate(startDate);

      //Add the record
      c.addRecord(record);
```

SOAP Request

```
<soapenv:Body>
        <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record xsi:type="ns8:BillingAccount" xmlns:ns8="urn:relationships_2017_1.lists.web
services.netsuite.com">
                <ns8:customer xsi:type="ns9:RecordRef" internalId="305" xmlns:ns9="urn:core_201
7_1.platform.webservices.netsuite.com"/>
                <ns8:name xsi:type="xsd:string">FedEx Account</ns8:name>
                <ns8:class xsi:type="ns10:RecordRef" internalId="1" xmlns:ns10="urn:core_2017_1
.platform.webservices.netsuite.com"/>
                <ns8:billingSchedule xsi:type="ns11:RecordRef" internalId="2" xmlns:ns11="urn:c
ore_2017_1.platform.webservices.netsuite.com"/>
```

```
                <ns8:frequency xsi:type="ns12:BillingAccountFrequency" xmlns:ns12="urn:types.re
lationships_2017_1.lists.webservices.netsuite.com">_monthly</ns8:frequency>
                <ns8:startDate xsi:type="xsd:dateTime">2017-01-16T02:43:47.987Z</ns8:startDate>

                <ns8:invoiceForm xsi:type="ns13:RecordRef" internalId="69" xmlns:ns13="urn:core
_2017_1.platform.webservices.netsuite.com"/>
            </record>
        </add>
    </soapenv:Body>
```

Billing Schedule

The billing schedule record enables you to create schedules that define how the bill for a transaction is relayed to the customer.

In general, a billing schedule determines the frequency with which the customer is billed and the amount of each bill. However, the exact effect of a billing schedule varies depending on its type. NetSuite includes five billing schedule types, all of which are supported in web services. For an overview of the types, their prerequisite features, and their entry forms in the user interface, see Billing Schedule Types.

The billing schedule record is defined in the listAcct (Accounting) XSD, where it is called BillingSchedule.

The following topics include more details on using SuiteTalk to work with billing schedules:

- Billing Schedule Supported Operations
- Billing Schedule Body Fields and Sublist Fields
- Billing Schedule Code Samples
- Common Errors with Billing Schedules

Billing Schedule Types

NetSuite includes five types of billing schedules. This topic summarizes some of the differences among the types.

Types and Their Required Features

The types available in your account are determined by the features you have enabled.

Type	Features Required for this Type
Standard	■ Advanced Billing
Charge-based	■ Advanced Billing ■ Project Management ■ Charge-Based Billing
Fixed bid interval	■ Advanced Billing ■ Project Management
Fixed bid milestone	
Time and materials	

Finding the Record in the User Interface

Before you begin work on your integration, it might be useful to view the billing schedule record in the user interface. The path to the record's entry form varies depending on the type of billing schedule you want to view. The layout and behavior of the entry form also vary depending on the type:

- Fixed Bid Milestone
- All Other Types

Fixed Bid Milestone

To display the entry form for a fixed bid milestone schedule, open a project record. Navigate to Financial subtab. Set the Billing Type dropdown list to Fixed Bid Milestone, and click the Plus icon button displayed next to the Billing Schedule field.

All Other Types

To see the entry forms used to create all other types of billing schedules, navigate to Lists > Accounting > Billing Schedules > New. You can choose among the different types using the Type dropdown list. If the Type dropdown list is not available, that means only one type of schedule is available in your account. To make more types available, enable more features. (For details, see Types and Their Required Features.)

More Information

For details on how to work with billing schedule records in the user interface, see the following topics:

- Creating Billing Schedules
- Items and Billing Schedules with Charge-Based Billing
- Creating a Fixed Bid, Interval Billing Schedule
- Creating a Milestone Billing Schedule
- Creating a Time and Materials Billing Schedule

Billing Schedule Body Fields and Sublist Fields

When you view the definition for the billing schedule record in the listAcct (Accounting) XSD, you see two sublists and quite a few body fields. However, the exact fields and sublists available vary depending on the type of schedule you are working with. This section highlights some differences but is not comprehensive. You may also want to refer to the entry forms available in the user interface, as described in Finding the Record in the User Interface.

Milestone Sublist

The Milestone sublist is available only when scheduleType is set to _fixedBidMilestone. The key for this sublist is milestoneId.

If you create lines in the Milestone sublist, omit any value for the initialAmount body field. When lines exist in the Milestone sublist, initialAmount is automatically populated. The reason is that with this type of schedule, initialAmount represents the difference between 100 percent and the total of the Milestone sublist lines.

If you omit lines in the Milestone sublist, you must set a value of "100%" for the initialAmount field.

For an example of how to populate the Milestone sublist when adding a record, see Adding a Fixed Bid Milestone Billing Schedule.

Recurrence Sublist

The recurrence sublist is available only when scheduleType is set to _standard and the frequency field is set to _custom. The key for this sublist is recurrenceId.

If you create lines in the recurrence sublist, you should omit any value for the numberRemaining body field. When lines exist in the rRecurrence sublist, the numberRemaining field is automatically populated. Note that numberRemaining is labeled Recurrence Count in the user interface.

For an example of how to populate the recurrence sublist when adding a record, see Adding a Standard Billing Schedule.

initialAmount

The initialAmount body field is available only with three types of billing record: standard, fixed bid interval, and fixed bid milestone:

- With fixed bid milestone billing schedules — If you are creating a fixed bid milestone type schedule, the initialAmount field always represents a percentage. Also, the behavior of this field varies depending on whether you create lines in the milestone sublist. For more details, see Milestone Sublist.
- With standard or fixed bid interval schedules — If you are creating a standard or fixed bid interval type record, it is technically possible to populate the initialAmount field with either a percentage or fixed value. However, unless you use line-level billing, you should populate this field with a percentage value only.

  If you do use line-level billing and want to set an initialAmount value that is fixed (for example, if you want the first bill to be $50), use an expression like the following:

```
// C#:
myBillingSchedule.initialAmount = "50";

// Java:
myBillingSchedule.setInitialAmount("50");
```

  To make the first bill a percentage of the total, use an expression like the following:

```
// C#
myBillingSchedule.initialAmount = "50%";

// Java:
myBillingSchedule.setInitialAmount("50%");
```

project

If you are adding a fixed bid milestone billing schedule, you must use the project body field to identify an existing project record. Otherwise, the add operation fails.

However, note that the new billing schedule is not automatically attached to the project record. To create a link between the project and the new billing schedule, you must update the project record. For an example, see Attaching a Billing Schedule to a Project.

For all other types, you do not have to include a project value (and if you do, your input is ignored).

More Information

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's billing schedule reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Billing Schedule Supported Operations

The following operations are supported for use with the billing schedule record:

add | addList | delete | deleteList | get | getList | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Note: For examples of how to use supported SuiteTalk operations with the billing schedule record, see Billing Schedule Code Samples.

Billing Schedule Code Samples

For examples of how to work with billing schedules using SuiteTalk, refer to the following sections:

- Adding a Standard Billing Schedule
- Adding a Fixed Bid Milestone Billing Schedule
- Attaching a Billing Schedule to an Item
- Attaching a Billing Schedule to a Project

Adding a Standard Billing Schedule

This example shows how to create a standard billing schedule. In this example, the frequency field is set to custom and recurrence sublist lines are created. (Both of these conditions are required to populate the recurrence sublist. For more details, see Recurrence Sublist.)

Because the request includes recurrence sublist lines, it is not necessary to set the numberRemaining body field. In fact, if you try to set a value for numberRemaining in this scenario, your input is ignored. (Note that in the user interface, the numberRemaining field is labeled Recurrence Count.)

Java

```
// Create billing schedule object.

BillingSchedule myBillingSchedule = new BillingSchedule();
```

```
// Set scheduleType and other needed body fields.

myBillingSchedule.setScheduleType(BillingScheduleType._standard);
myBillingSchedule.setFrequency(BillingScheduleFrequency._custom);
myBillingSchedule.setName("With recurrence sublist");
myBillingSchedule.setExternalId("1201A");
myBillingSchedule.setInitialAmount("500.00");


// Create the sublist and populate it with two lines.

BillingScheduleRecurrenceList myBillingScheduleRecurrenceList = new BillingScheduleRecurrenceLi
st();
BillingScheduleRecurrence[] myBillingScheduleRecurrence = new BillingScheduleRecurrence[2];

myBillingScheduleRecurrence[0] = new BillingScheduleRecurrence();
myBillingScheduleRecurrence[0].setCount(3L);
myBillingScheduleRecurrence[0].setUnits(BillingScheduleRecurrenceRecurrenceUnits._months);
myBillingScheduleRecurrence[0].setAmount(10.0);

myBillingScheduleRecurrence[1] = new BillingScheduleRecurrence();
myBillingScheduleRecurrence[1].setCount(2L);
myBillingScheduleRecurrence[1].setUnits(BillingScheduleRecurrenceRecurrenceUnits._weeks);
myBillingScheduleRecurrence[1].setAmount(15.0);

myBillingScheduleRecurrenceList.setBillingScheduleRecurrence(myBillingScheduleRecurrence);
myBillingSchedule.setRecurrenceList(myBillingScheduleRecurrenceList);


// Execute the operation.

ns._port.add(myBillingSchedule);
```

SOAP Request

```
<soapenv:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record xsi:type="ns1:BillingSchedule" externalId="1201A" xmlns:ns1="urn:accounting_2017_
1.lists.webservices.netsuite.com">
         <ns1:scheduleType xsi:type="ns2:BillingScheduleType" xmlns:ns2="urn:types.accounting_2
017_1.lists.webservices.netsuite.com">_standard</ns1:scheduleType>
         <ns1:name xsi:type="xsd:string">With recurrence sublist</ns1:name>
         <ns1:initialAmount xsi:type="xsd:string">500.00</ns1:initialAmount>
         <ns1:frequency xsi:type="ns3:BillingScheduleFrequency" xmlns:ns3="urn:types.accounting
_2017_1.lists.webservices.netsuite.com">_custom</ns1:frequency>
         <ns1:recurrenceList xsi:type="ns1:BillingScheduleRecurrenceList" replaceAll="false">
            <ns1:billingScheduleRecurrence xsi:type="ns1:BillingScheduleRecurrence">
               <ns1:count xsi:type="xsd:long">3</ns1:count>
               <ns1:units xsi:type="ns4:BillingScheduleRecurrenceRecurrenceUnits" xmlns:ns4="ur
n:types.accounting_2017_1.lists.webservices.netsuite.com">_months</ns1:units>
               <ns1:amount xsi:type="xsd:double">10.0</ns1:amount>
            </ns1:billingScheduleRecurrence>
            <ns1:billingScheduleRecurrence xsi:type="ns1:BillingScheduleRecurrence">
```

```
               <ns1:count xsi:type="xsd:long">2</ns1:count>
               <ns1:units xsi:type="ns5:BillingScheduleRecurrenceRecurrenceUnits" xmlns:ns5="ur
n:types.accounting_2017_1.lists.webservices.netsuite.com">_weeks</ns1:units>
               <ns1:amount xsi:type="xsd:double">15.0</ns1:amount>
            </ns1:billingScheduleRecurrence>
         </ns1:recurrenceList>
      </record>
   </add>
</soapenv: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="billingSchedule" externalId="1201A" i
nternalId="98" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Adding a Fixed Bid Milestone Billing Schedule

The following example shows how to create a fixed bid milestone billing schedule. In this example, lines are created in the milestone sublist. (The only way to populate the milestone sublist is when creating a fixed bid milestone billing schedule. For more details, see Milestone Sublist.)

The milestone sublist determines when bills will be sent. Each line can reference a date that you manually specify or an existing milestone task record.

If you want your sublist lines to reference existing milestone tasks, use code like the following for each sublist line:

```
// In C#:

RecordRef myMilestoneTask1 = new RecordRef();
myMilestoneTask1.type = RecordType.projectTask;
myMilestoneTask1.typeSpecified = true;
myMilestoneTask1.internalId = "3425";
myBillingSchedule.milestoneList.billingScheduleMilestone[0] = new BillingScheduleMilestone();
myBillingSchedule.milestoneList.billingScheduleMilestone[0].projectTask = myMilestoneTask1;
myBillingSchedule.milestoneList.billingScheduleMilestone[0].milestoneAmount = 45;
myBillingSchedule.milestoneList.billingScheduleMilestone[0].milestoneAmountSpecified = true;
```

Any milestone task you reference must have as its parent the project identified in the billing schedule's project body field. In other words, you can't reference Project A in the project body field and then list milestone tasks from Project B in the sublist. If you do, the add operation fails.

You can also manually choose a date for each line in the milestone sublist, as shown in the following sample.

See also Milestone Sublist.

Note: Be aware that even after you create a billing schedule, it is not automatically attached to any project record (even though you had to reference one when creating the schedule). You must manually attach the billing schedule to the project record. For details, see Attaching a Billing Schedule to a Project.

C#

```
private void addFixedBidMilestoneBillingSchedule()
{

   // Create object.

   BillingSchedule myBillingSchedule = new BillingSchedule();


   // Set an external Id.

   myBillingSchedule.externalId = "103A";


   // Set the billing schedule type.

   myBillingSchedule.scheduleType = BillingScheduleScheduleType._fixedBidMilestone;
   myBillingSchedule.scheduleTypeSpecified = true;


   // Identify an existing project recorded listed at
   // Lists > Relationships > Projects.

   RecordRef projectRef = new RecordRef();
   projectRef.internalId = "1461";
   projectRef.type = RecordType.job;
   projectRef.typeSpecified = true;
   myBillingSchedule.project = projectRef;


   // Give the schedule a name.

   myBillingSchedule.name = "Milestone Schedule - w/Fixed Dates";


   // Create a sublist that can accommodate three lines

   myBillingSchedule.milestoneList = new BillingScheduleMilestoneList();
   myBillingSchedule.milestoneList.billingScheduleMilestone = new BillingScheduleMilestone[3];


   // Populate each line with a manually defined date and billing percentage.

   DateTime nextWeek = new DateTime(2014, 4, 5);
   myBillingSchedule.milestoneList.billingScheduleMilestone[0] = new BillingScheduleMilestone()
;
   myBillingSchedule.milestoneList.billingScheduleMilestone[0].milestoneDate = nextWeek;
   myBillingSchedule.milestoneList.billingScheduleMilestone[0].milestoneDateSpecified = true;
   myBillingSchedule.milestoneList.billingScheduleMilestone[0].milestoneAmount = 25;
   myBillingSchedule.milestoneList.billingScheduleMilestone[0].milestoneAmountSpecified = true;


   DateTime inTwoWeeks = new DateTime(2014, 4, 19);
   myBillingSchedule.milestoneList.billingScheduleMilestone[1] = new BillingScheduleMilestone()
;
   myBillingSchedule.milestoneList.billingScheduleMilestone[1].milestoneDate = inTwoWeeks;
   myBillingSchedule.milestoneList.billingScheduleMilestone[1].milestoneDateSpecified = true;
   myBillingSchedule.milestoneList.billingScheduleMilestone[1].milestoneAmount = 25;
   myBillingSchedule.milestoneList.billingScheduleMilestone[1].milestoneAmountSpecified = true;


   DateTime nextMonth = new DateTime(2014, 5, 5);
   myBillingSchedule.milestoneList.billingScheduleMilestone[2] = new BillingScheduleMilestone()
;
   myBillingSchedule.milestoneList.billingScheduleMilestone[2].milestoneDate = nextMonth;
   myBillingSchedule.milestoneList.billingScheduleMilestone[2].milestoneDateSpecified = true;
   myBillingSchedule.milestoneList.billingScheduleMilestone[2].milestoneAmount = 25;
   myBillingSchedule.milestoneList.billingScheduleMilestone[2].milestoneAmountSpecified = true;



   // Execute the add operation.

   _service.add(myBillingSchedule);

}
```

SOAP Request

```
<soap:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record xmlns:q1="urn:accounting_2017_1.lists.webservices.netsuite.com" xsi:type="q1:Bill
ingSchedule" externalId="103A">
         <q1:scheduleType>_fixedBidMilestone</q1:scheduleType>
         <q1:name>Milestone Schedule - w/Fixed Dates</q1:name>
         <q1:project internalId="1461" type="job" />
         <q1:milestoneList>
            <q1:billingScheduleMilestone>
               <q1:milestoneAmount>25</q1:milestoneAmount>
               <q1:milestoneDate>2014-04-05T00:00:00</q1:milestoneDate>
         </q1:billingScheduleMilestone>
            <q1:billingScheduleMilestone>
               <q1:milestoneAmount>25</q1:milestoneAmount>
               <q1:milestoneDate>2014-04-19T00:00:00</q1:milestoneDate>
         </q1:billingScheduleMilestone>
            <q1:billingScheduleMilestone>
               <q1:milestoneAmount>25</q1:milestoneAmount>
               <q1:milestoneDate>2014-05-05T00:00:00</q1:milestoneDate>
            </q1:billingScheduleMilestone>
         </q1:milestoneList>
      </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 internalId="34" externalId="103A" type="billingSchedule" xsi:type="platformCo
re:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Attaching a Billing Schedule to an Item

After you create a billing schedule, you may want to attach it to an item record. In general, when you attach a billing schedule to an item record, the schedule is the default chosen when that item is added to a sales order.

For more information about attaching billing schedules to item records, see the help topic Applying Billing Schedules.

Note: You can also attach billing schedules to project records. See Attaching a Billing Schedule to a Project.

The following example shows how to attach a billing schedule to an inventory item.

Java

```
{

   // Identify the inventory item to which you
   // want to attach the billing schedule.

   InventoryItem myItem = new InventoryItem();
   myItem.setInternalId("552");


   // Identify the billing schedule you want to use.

   RecordRef myBillingSchedule = new RecordRef();
   myBillingSchedule.setInternalId("1");
   myItem.setBillingSchedule(myBillingSchedule);


   // Execute the update operation.

   ns._port.update(myItem);

}
```

SOAP Request

```
<soapenv:Body>
   <update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record xsi:type="ns1:InventoryItem" internalId="552" xmlns:ns1="urn:accounting_2017_1.li
sts.webservices.netsuite.com">
         <ns1:billingSchedule xsi:type="ns2:RecordRef" internalId="1" xmlns:ns2="urn:core_2017_
1.platform.webservices.netsuite.com"/>
      </record>
   </update>
</soapenv:Body>
```

SOAP Response

```
<soapenv:Body>
   <updateResponse 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" internalId="552" xmlns
:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </updateResponse>
</soapenv:Body>
```

Attaching a Billing Schedule to a Project

In some cases, you may want to attach a billing schedule to a project record.

The type of billing schedule you choose must match the project record's billingType body field. If you try to attach a different type of billing schedule, the system generates an error. For this reason, you may want to specify the billing type as part of the same update operation that attaches the schedule to the project record.

If you used SuiteTalk to create a fixed bid milestone billing schedule, you may recall that you were required to specify a project record as part of the add operation. However, the billing schedule was not automatically applied to the project record as part of the add operation. You must manually attach the schedule to the project record. If the billing schedule references milestone task records associated with a particular project, the schedule can be attached only to that project.

For more information on working with billing schedules in conjunction with projects, see the help topic Project Billing.

C#

```
private void attachScheduleToProject()
{

   // Define the project to which you want to
   // attach the billing schedule.

   Job updatedProject = new Job();
   updatedProject.internalId = "1461";
```

```
    // To avoid a potential mismatch with the billing
    // type field, you may want to actively set it to
    // the desired type.

    updatedProject.jobBillingType = JobBillingType._fixedBidMilestone;
    updatedProject.jobBillingTypeSpecified = true;


    // Identify the billing schedule.

    RecordRef myBillingSchedule = new RecordRef();
    myBillingSchedule.type = RecordType.billingSchedule;
    myBillingSchedule.typeSpecified = true;
    myBillingSchedule.internalId = "34";


    // Update the project record so that it uses
    // this billing schedule.

    updatedProject.billingSchedule = myBillingSchedule;


    // Execute the update operation.

    _service.update(updatedProject);

}
```

SOAP Request

```
<soap:Body>
   <update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record xmlns:q1="urn:relationships_2017_1.lists.webservices.netsuite.com" xsi:type="q1:J
ob" internalId="1461">
         <q1:jobBillingType>_fixedBidMilestone</q1:jobBillingType>
         <q1:billingSchedule internalId="34" type="billingSchedule" />
      </record>
   </update>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
   <updateResponse 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 internalId="1461" type="job" xsi:type="platformCore:RecordRef" xmlns:platform
Core="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </updateResponse>
</soapenv:Body>
```

Common Errors with Billing Schedules

When you are working with billing schedule records, the following failure notices can occur in your SOAP responses.

A record with the same unique signatures already exists

The system generates this error when you try to create a new billing schedule with an external Id that is already in use. Either change the external Id of the record you are adding, or make a change to the existing record.

Invalid projecttask reference key x

This error can occur when you are trying to create a fixed bid milestone billing schedule. In some cases, you might create a sublist that references existing milestone tasks. If you use milestone tasks that do not match the project identified through the project body field, the system generates this error.

Invalid billingschedule reference key x for jobbillingtype y

You might see this error when attempting to attach a billing schedule to a project record. This error relates to the project record's Billing Type field. This field must be set to the correct type for the schedule you are trying to attach. If Billing Type is set to a different type, or if it has not been set, the system returns this error.

Recurring monthly events cannot be scheduled after the 28th

You may see this error when both of the following are true:

- You are trying to add a charge-based billing schedule with the frequency body field set to _monthly.
- The seriesStartDate body field is set to the 28th, the 29th, the 30th, or the 31st of any month, or the field is left empty but the SOAP request is sent on one of those dates.

You must choose a recurrence count greater than 0

This message refers to a problem with the numberRemaining field (numberRemaining is labeled Recurrence Count in the user interface). The message indicates that you failed to set a value for the field. The field is required when frequency is set to any value other than _never.

Bin

To use Bins the Bin Management feature must be enabled at Setup > Company > Setup Tasks > Enable Features. On the Items & Inventory tab, check the Bin Management box.

The bin record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the bin record.

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's bin reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Currently there are no available search joins for this record.

Classification (Class)

The classification record is defined in the listAcct (accounting) XSD.

For information about working with classes in the UI, see the help topic Classifications in NetSuite.

Supported Operations

The following operations can be used with the classification (class) record.

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's classification reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Consolidated Exchange Rate

If you use NetSuite OneWorld and your subsidiaries have different base currencies, you can maintain a table of consolidated exchange rates. This table is used to ensure that for consolidation purposes, currency amounts properly roll up from child to parent subsidiaries. Each consolidated exchange rate translates between the base currency of a subsidiary and the base currency of its parent or grandparent subsidiary, for a specified accounting period. Consolidated exchange rates include three different rate types per period and subsidiary pair: Current, Average, and Historical.

The Consolidated Exchange Rates table is available at Lists > Accounting > Consolidated Exchange Rates (Administrator). Consolidated exchange rates for each period are based on the currency exchange rates effective at the end of the period, and on rates used for transactions posted during the period. You should update consolidated exchange rates at the end of each accounting period, as part of period close tasks.

For more information about the consolidated exchange rate record and working with it in the user interface, see the help topic Using Consolidated Exchange Rates.

Consolidated exchange rates cannot be created or deleted in web services, they may only be viewed and edited. For more information, see Consolidated Exchange Rate Supported Operations for the consolidated exchange rate record.

The consolidated exchange rate record is defined in the listAcct (accounting) XSD, where it is called ConsolidatedExchangeRate.

The following topics include more details on using SuiteTalk to work with consolidated exchange rates:

- Consolidated Exchange Rate Supported Operations
- Consolidated Exchange Rate Field Definitions
- Consolidated Exchange Rate Usage Notes
- Consolidated Exchange Rate Code Samples

Consolidated Exchange Rate Supported Operations

The following operations can be used with the consolidated exchange rate record.

get |getList | getSavedSearch | search | searchMore | searchMoreWithId | searchNext | update | updateList |

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Note: Direct rates, between child and parent subsidiaries are editable for periods that have not been closed (that have a value of No in the Closed column). Indirect rates, also known as implied rates, between subsidiaries more than one hierarchical level removed from each other are never editable. Additionally, you cannot edit rates for elimination subsidiaries.

Consolidated Exchange Rate Field Definitions

When editing a consolidated exchange rate record, you are required to define the following body fields:

Field Name	Label in UI	Description
`averageExchangeRate`	Average Exchange Rate	The average exchange rate. This rate is calculated from a weighted average of the exchange rates for transactions applied during the period to accounts

Field Name	Label in UI	Description
		with a general rate type of Average. This rate is used to translate accounts in the income statement and to build retained earnings. This is a double field.
`currentExchangeRate`	Current Exchange Rate	The current exchange rate. Also referred to as ending rate. This rate is based on the currency exchange rate that is effective at the end of the reported upon period. This rate is used for most asset and liability accounts in the balance sheet. This is a double field.
`historicalExchangeRate`	Historical Exchange Rate	The historical exchange rate. This rate is calculated from a weighted average of the exchange rates for transactions applied during the period to accounts with a general rate type of Historical. This rate is used for equity accounts, or owner's investments, in the balance sheet. This is a double field.

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's consolidated exchange rate reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Consolidated Exchange Rate Usage Notes

Your user role must have the Currency permission with the Full permission level to work with consolidated exchange rates in the user interface and in web services.

Consolidated Exchange Rate Code Samples

For examples of how to work with consolidated exchange rate records using SuiteTalk, refer to the following sections:

- Getting a Consolidated Exchange Rate Record
- Updating a Consolidated Exchange Rate Record

Getting a Consolidated Exchange Rate Record

This example shows how to access a consolidated exchange rate record using the get operation.

Java

```
public void testGETConsolidatedExchangeRate() throws Exception
{
c.useRequestLevelCredentials()
c.getRecord("11445", RecordType.consolidatedExchangeRate)
}
```

SOAP Request

```
        <soapenv:Body>
         <get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
             <baseRef xsi:type="ns7:RecordRef" type="consolidatedExchangeRate" internalId="11445
" xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com"/>
         </get>
     </soapenv:Body>
```

SOAP Response

```
         <soapenv:Body>
         <getResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
             <readResponse>
                 <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platf
orm.webservices.netsuite.com"/>
                 <record xsi:type="listAcct:ConsolidatedExchangeRate" internalId="11445" xmlns:l
istAcct="urn:accounting_2017_1.lists.webservices.netsuite.com">
                     <listAcct:postingPeriod>420</listAcct:postingPeriod>
                     <listAcct:fromSubsidiary>5</listAcct:fromSubsidiary>
                     <listAcct:fromCurrency>4</listAcct:fromCurrency>
                     <listAcct:toSubsidiary>3</listAcct:toSubsidiary>
                     <listAcct:toCurrency>1</listAcct:toCurrency>
                     <listAcct:averageRate>1.06882501</listAcct:averageRate>
                     <listAcct:currentRate>1.06882501</listAcct:currentRate>
                     <listAcct:historicalRate>1.06882501</listAcct:historicalRate>
                     <listAcct:accountingBook>1</listAcct:accountingBook>
                     <listAcct:isPeriodClosed>false</listAcct:isPeriodClosed>
                     <listAcct:isDerived>false</listAcct:isDerived>
                     <listAcct:isEliminationSubsidiary>false</listAcct:isEliminationSubsidiary>
                 </record>
             </readResponse>
         </getResponse>
     </soapenv:Body>
```

Updating a Consolidated Exchange Rate Record

This example shows how to update a consolidated exchange rate record using the update operation.

Java

```
public void testUPDATEConsolidatedExchangeRate () throws Exception
{
      c.useRequestLevelCredentials()

      ConsolidatedExchangeRate consolidatedExchangeRate = new ConsolidatedExchangeRate()
      consolidatedExchangeRate.setInternalId("11445")
      consolidatedExchangeRate.setAverageRate(1.06882501)
      consolidatedExchangeRate.setCurrentRate(1.06882501)
      consolidatedExchangeRate.setHistoricalRate(1.06882501)

c.updateRecord(consolidatedExchangeRate)
}
```

SOAP Request

```
<update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
        <record internalId="11445" xsi:type="ns7:ConsolidatedExchangeRate" xmlns:ns7="urn:accou
nting_2017_1.lists.webservices.netsuite.com">
                  <ns7:averageRate xsi:type="xsd:double">10.0</ns7:averageRate>
             <ns7:currentRate xsi:type="xsd:double">10.0</ns7:currentRate>
            <ns7:historicalRate xsi:type="xsd:double">10.0</ns7:historicalRate>
      </record>
</update>
```

SOAP Response

```
<updateResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <writeResponse>
                 <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.plat
form.webservices.netsuite.com"/>
               <baseRef internalId="11445" type="consolidatedExchangeRate" xsi:type="platformCo
re:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
</updateResponse>
```

Currency

The currency record is defined in the listAcct (accounting) XSD.

Note that you cannot use web services to the get exchange rates specified on currency records. In both the UI and in web services, the exchangeRate field is a create-time-only field that is used to specify the initial exchange rate with respect to the base currency. After the inital exchange rate value is set, this value does not get updated to reflect real-time exchange rates. Therefore, if you were to attempt to get “today's” current exchange rate, the value returned could be inaccurate.

Supported Operations

The following operations can be used with the currency record.

add | addList | delete | deleteList | get | getAll | getList | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s currency reference page.

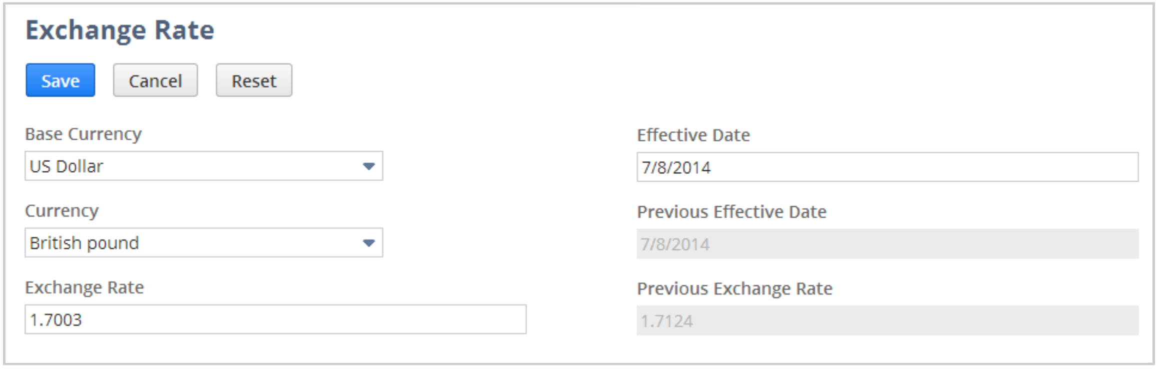
Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Currency Rate

The currency rate record is defined in the listAcct (accounting) XSD.

The currency rate record is the same record that appears as the **Exchange Rate** record in the UI (see figure). In the UI, access this record by going to Lists > Accounting > Currency Exchange Rates > New.

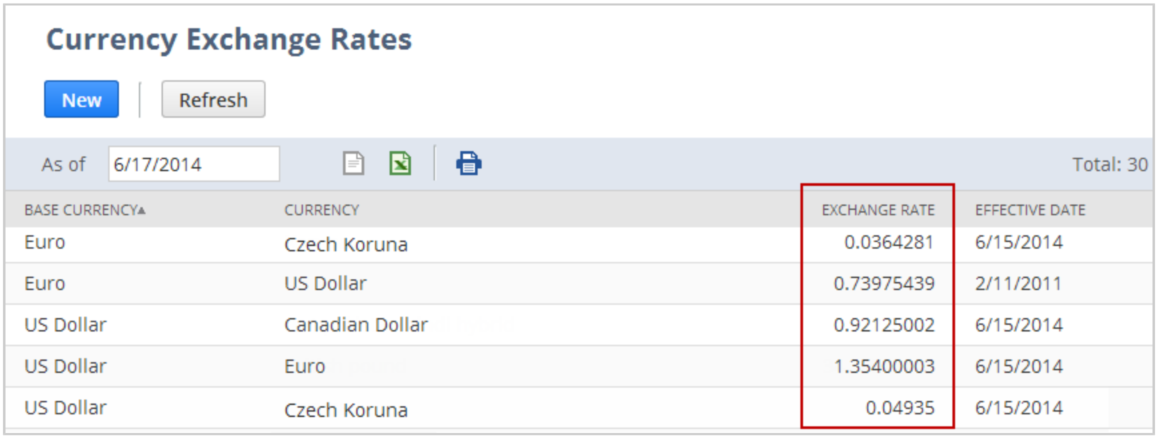
Note: The Multiple Currencies feature must be enabled in your account before using this record. For information on enabling this feature, see the help topic Enabling the Multiple Currencies Feature.



Base Currency: US Dollar	Effective Date: 7/8/2014
Currency: British pound	Previous Effective Date: 7/8/2014
Exchange Rate: 1.7003	Previous Exchange Rate: 1.7124

In web services, you can add new currency rate records, but you cannot update them. That is, after added the record, you cannot use any of the update operations to update the record. Update operations include update, updateList, upsert, and upsertList.

After adding a currency rate record, your data will appear in the Currency Exchange Rates table (see below).



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

In web services you must use the getCurrencyRate operation to get the exchange rate between two currencies based on a certain date. (See the help topic getCurrencyRate for more information on using this operation.)

Note that in both the UI and in web services, the Exchange Rate (exchangeRate) field is a create-time-only field that is used to specify the initial exchange rate with respect to the base currency. After the initial exchange rate value is set, this value does not get updated to reflect real-time exchange rates. Therefore, if you were to attempt to get today's current exchange rate, the value returned could be inaccurate.

Note: The Previous Effective Date and the Previous Exchange Rate fields are not currently available in web services.

Supported Operations

The following operations can be used with the currency rate record.

add | addList | delete | deleteList | get | getList | getCurrencyRate | search | getDeleted

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's currency reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

The follow sample shows how to add a currency rate record through SuiteTalk.

Code Sample

Java

```
 // USD -> GBP, from tomorrow, with rate 1.4488
public void addCurrencyRate() throws Exception
{
   this.login();

   RecordRef usd = new RecordRef();
   usd.setInternalId("1");
   usd.setType(RecordType.currency);

   RecordRef gbp = new RecordRef();
   gbp.setInternalId("2");
   gbp.setType(RecordType.currency);

   com.netsuite.devtools.lists.accounting.CurrencyRate crs = new com.netsuite.devtools.lists
.accounting.CurrencyRate();
      crs.setBaseCurrency(usd);
      crs.setTransactionCurrency(gbp);
      crs.setExchangeRate(1.4488D);

      Calendar cal = Calendar.getInstance();
      cal.add(Calendar.DAY_OF_YEAR, +1);   //tomorrow
      crs.setEffectiveDate(cal);

      _port.add(crs);
   }
```

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>
        <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record xsi:type="ns6:CurrencyRate" xmlns:ns6="urn:accounting_2017_1.lists.webservi
ces.netsuite.com">
                <ns6:baseCurrency xsi:type="ns7:RecordRef" type="currency" internalId="1" xmlns
:ns7="urn:core_2017_1.platform.webservices.netsuite.com"/>
                <ns6:transactionCurrency xsi:type="ns8:RecordRef" type="currency" internalId="2
" xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com"/>
                <ns6:exchangeRate xsi:type="xsd:double">1.4488</ns6:exchangeRate>
                <ns6:effectiveDate xsi:type="xsd:dateTime">2012-08-11T06:50:09.038Z</ns6:effect
iveDate>
            </record>
        </add>
    </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_080920122566427461899365710_9bfa26bd40b8</pl
atformMsgs:nsId>
        </platformMsgs:documentInfo>
    </soapenv:Header>
    <soapenv:Body>
        <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <writeResponse>
                <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platf
orm.webservices.netsuite.com"/>
                <baseRef xsi:type="platformCore:RecordRef" type="currencyRate" internalId="58"
xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </writeResponse>
        </addResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Department

The department record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the department record.

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's department reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Expense Category

The expense category record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the expense category record.

add | addList | delete | deleteList | get | getAll | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s expense category reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Before you begin working with expense categories, you need to understand the rateRequired field. The purpose of this field is to determine the behavior of expenses that use this category. That is, when rateRequired is set to true, an expense report line item using this category must have values in the Rate and Quantity fields. These values are then used to determine the amount of the expense. Note that you can also set default rates for your expense categories. You do this by using either the defaultRate body field (in accounts that are not OneWorld) or the Rates sublist (in OneWorld accounts).

Related to your ability to create defaults, be aware of the following: If your account is not a OneWorld account, the rateRequired value affects the availability of the defaultRate field. That is, when the value of rateRequired is true for any particular expense category, you can set a value for defaultRate for that category. Otherwise, the field is not available. The defaultRate field determines the default rate for the category throughout your organization.

If your account is a OneWorld account, you use the Rates sublist, which stores a default rate for the category for each subsidiary. Unlike the defaultRate body field, an expense category’s Rates sublist is available even when you set the rateRequired value to false.

The default value of rateRequired is false.

Fair Value Price

The fair value price record is part of the Advanced Revenue Management feature. The FairValuePrice complex type is defined in the Accounting XSD.

Before you can use web services to create fair value price records, you must have enabled the Advanced Revenue Management feature. You must also have added a formula for calculating a fair value price. For general information on the Advanced Revenue Management feature, see the help topic Advanced Revenue Management. For details about the fair value price record, see the help topic Fair Value Setup.

Configuration of advanced revenue management is complex. NetSuite strongly recommends that you engage NetSuite Professional Services or a NetSuite partner to assist with your implementation.

Supported Operations

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's fair value price reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

In the following example, a fair value price record is updated with new values.

Java

```
public void FairValuePrice() throws Exception {

    RecordRef currencyRef = new RecordRef();
    currencyRef.setInternalId("1");

    FairValuePrice fairValue = new FairValuePrice();
    fairValue.setInternalId("1");
    fairValue.setExternalId("FairValue_001");
    fairValue.setCurrency(currencyRef);
    fairValue.setFairValueFormula(FairValueFormula);
    fairValue.setFairValue(new Double("200"));
    fairValue.setLowValuePercent(new Double("80"));
    fairValue.setHighValuePercent(new Double("115"));

    c.updateRecord(FairValue);

}
```

SOAP Request

```
 <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:applicationInfo soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next" soapen
v:mustUnderstand="0" xmlns:ns1="urn:messages_2017_1.platform.webservices.netsuite.com">
```

```
            <ns1:applicationId>APPLICATIONS_ID</ns1:applicationId>
         </ns1:applicationInfo>
         <ns2:passport soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next" soapenv:mustU
nderstand="0" xmlns:ns2="urn:messages_2017_1.platform.webservices.netsuite.com">

            <ns3:email xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">EMAIL</ns3
:email>
            <ns4:password xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com">PASSWOR
D</ns4:password>
            <ns5:account xmlns:ns5="urn:core_2017_1.platform.webservices.netsuite.com">ACCOUNT_
ID</ns5:account>
            <ns6:role internalId="3" xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.c
om"/>
         </ns2:passport>
      </soapenv:Header>
      <soapenv:Body>
         <update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record externalId="FairValue_001" internalId="1" xsi:type="ns7:FairValuePrice" xml
ns:ns7="urn:accounting_2017_1.lists.webservices.netsuite.com">
               <ns7:fairValue xsi:type="xsd:double">200.0</ns7:fairValue>
               <ns7:currency internalId="1" xsi:type="ns8:RecordRef" xmlns:ns8="urn:core_2017_1
.platform.webservices.netsuite.com"/>
               <ns7:lowValuePercent xsi:type="xsd:double">85.0</ns7:lowValuePercent>

               <ns7:highValuePercent xsi:type="xsd:double">115.0</ns7:highValuePercent>

            </record>
         </update>
      </soapenv:Body>
   </soapenv:Envelope>
```

SOAP Response

```
   <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_xxxxxx_xxxxxxxxxxxxxxxx_xxxxxxxxx</platformMsgs:nsI
d>
         </platformMsgs:documentInfo>
      </soapenv:Header>
      <soapenv:Body>
         <updateResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <writeResponse>
               <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
               <baseRef internalId="1" externalId="FairValue_001" type="fairValuePrice" xsi:typ
e="platformCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.co
m"/>
            </writeResponse>
         </updateResponse>
      </soapenv:Body>
```

```
    </soapenv:Envelope>
```

Gift Certificate

A gift certificate record is created when a customer purchases a gift certificate item. For example, you might create a gift certificate item and make it available for purchase. A customer can then buy this gift certificate item. When the customer buys the gift certificate item and names a recipient, the system creates a gift certificate, which can be redeemed for goods and services sold by your business. The gift certificate includes properties such as an expiration date, a remaining value, and the email address of the gift certificate's recipient.

The gift certificate record is defined in the listAcct (accounting) XSD, where it is called giftCertificate.

To view existing gift certificate instances in the UI, go to Lists > Accounting > Gift Certificates.

For details about working with gift certificate items in the UI, see the help topic Viewing and Editing Gift Certificate Status. For full details about the gift certificate feature, see the help topic Gift Certificates.

The gift certificate item record is also supported in web services. For details, see Gift Certificate Item.

Supported Operations

The following operations can be used with the gift certificate record:

get | getList | getSavedSearch | search | searchMore | searchMoreWithId | searchNext | update | updateList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Note that when using the **update** operation, only the following fields can be updated:

- email
- expirationDate
- message
- name
- sender

Gift Certificate Field Definitions

This record does not support external ID.

For full details on all body fields, sublist fields, search filters, and search joins available to this record, see the Schema Browser's gift certificate reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Code Sample

The following example show how to update a gift certificate instance's with new values for the recipient, name, message, and expirationDate fields.

C#

```
private void updateGiftCertificate()
{
   GiftCertificate myGiftCertificate = new GiftCertificate();
   myGiftCertificate.internalId = "13";
   myGiftCertificate.email = "john@smith.com";
   myGiftCertificate.name = "John";
   myGiftCertificate.message = "Happy Birthday!";

   DateTime nextYear = new DateTime(2016, 1, 1);
   myGiftCertificate.expirationDate = nextYear;
   myGiftCertificate.expirationDateSpecified = true;

   _service.update(myGiftCertificate);

}
```

SOAP Request

```
<soap:Body>
   <update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record internalId="13" xsi:type="q1:GiftCertificate" xmlns:q1="urn:accounting_2017_1.lis
ts.webservices.netsuite.com">
         <q1:name>John</q1:name>
         <q1:email>john@smith.com</q1:email>
         <q1:message>Happy Birthday!</q1:message>
         <q1:expirationDate>2016-01-01T00:00:00</q1:expirationDate>
      </record>
   </update>
</soap:Body>
```

SOAP Response:

```
<soapenv:Body>
   <updateResponse 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="giftCertificate" internalId="13" xmln
s:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </updateResponse>
</soapenv:Body>
```

Inventory Number

The term inventory number refers to a serial number or a lot number. An inventory number record uniquely identifies an item in physical inventory with a serial number, or uniquely identifies a group of items with a lot number. For details about these types of items, see the help topics Serial Numbered

Items and Lot Numbered Items. This record is available when the Serialized Inventory or Lot Tracking feature is enabled.

The inventory number record is defined in the listAcct (accounting) XSD.

Important: You cannot create standalone inventory number records. The system generates these records when a serialized item or lot numbered item is created, based on the serial number or lot number values entered for the item record. See the help topics Adding New Serial Numbers to Inventory and Creating Lot Numbered Items. Also, you cannot delete inventory number records.

Supported Operations

The following operations can be used with the inventory number record.

get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's inventory number reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Item Revision

The exact member components needed for assembly items are identified in the Bill of Materials (BOM), but required components may change over time. These changing requirements can be documented in item revision records, which are part of BOM member control functionality. Item revision records define which member items should be included in assembly builds during specific time frames.

An item revision record sets an effective date for a member item to be included in assembly builds. Each item revision record can be assigned to multiple assembly items, because one item can be a member of different assembly items.

For more details about this functionality, see the help topic Bill of Materials Member Control for Assembly Items.

This record is available when the Assembly Items feature is enabled.

The item revision record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the item revision record.

add | addList | delete | deleteList | get | getList | search | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's item revision reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

The obsoleteDate field is read-only.

Code Samples

The following code adds an item revision.

SOAP Request

```
<soapenv:Body>
  <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <record xsi:type="ns6:ItemRevision" xmlns:ns6="urn:accounting_2017_1.lists.webservices.netsu
ite.com">
    <ns6:item internalId="109" type="assemblyItem" xsi:type="ns7:RecordRef" xmlns:ns7="urn:core
_2017_1.platform.webservices.netsuite.com"/>
    <ns6:name xsi:type="xsd:string">WS added revision</ns6:name>
    <ns6:effectiveDate xsi:type="xsd:dateTime">2012-03-06T23:00:00.000Z</ns6:effectiveDate>
    <ns6:memo xsi:type="xsd:string">WS added memo</ns6:memo>
    <ns6:inactive xsi:type="xsd:boolean">true</ns6:inactive>
   </record>
  </add>
</soapenv: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.platfo
rm.webservices.netsuite.com"/>
               <baseRef internalId="3" type="itemRevision" xsi:type="platformCore:RecordRef" xm
lns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
```

```
            </writeResponse>
         </addResponse>
      </soapenv:Body>
```

The following code gets item revision data.

SOAP Request

```
 <soapenv:Body>
         <get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <baseRef internalId="3" type="itemRevision" xsi:type="ns6:RecordRef" xmlns:ns6="urn
:core_2017_1.platform.webservices.netsuite.com"/>
         </get>
      </soapenv:Body>
```

SOAP Response

```
<soapenv:Body>
         <getResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <readResponse>
               <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
               <record internalId="3" xsi:type="listAcct:ItemRevision" xmlns:listAcct="urn:acco
unting_2017_1.lists.webservices.netsuite.com">
                  <listAcct:item internalId="109" xmlns:platformCore="urn:core_2017_1.platform.
webservices.netsuite.com">
                     <platformCore:name>assembly test item</platformCore:name>
                  </listAcct:item>
                  <listAcct:name>WS added revision</listAcct:name>
                  <listAcct:effectiveDate>2012-03-06T00:00:00.000-08:00</listAcct:effectiveDate
>

                  <listAcct:memo>WS added memo</listAcct:memo>
                  <listAcct:inactive>true</listAcct:inactive>
               </record>
            </readResponse>
         </getResponse>
      </soapenv:Body>
```

Location

The location record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the location record.

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's location reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Manufacturing Cost Template

If the Manufacturing Routing and Work Center feature has been enabled, you can use SuiteTalk to interact with manufacturing cost template records. You can check to see whether the feature is enabled by going to Setup > Company > Enable Features, and reviewing the Items & Inventory tab.

The benefit of this feature is that it lets you specify a sequence of tasks required for the completion of a Work In Process (WIP) work order. You take advantage of this feature using a few records, including cost template. The purpose of the cost template record is to group together the various expenses associated with a specific activity. These expenses might include the cost of paying employees, operating machinery, and so on.

For details on the process of manually creating a cost template, see the help topic Creating Manufacturing Cost Templates. For more information about the Manufacturing Routing and Work Center feature, see the help topic Manufacturing Routing.

The manufacturing cost template record is defined in the listScm XSD.

Supported Operations

The following operations can be used with the manufacturing cost template record.

add | addList | delete | deleteList | get |getDeleted |getList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's manufacturing cost template reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Refer to the following sections for more details on interacting with manufacturing cost template records.

Prerequisites for Adding a Record

When you create a cost template record, at least one row is required in the costDetail sublist. Each sublist entry must reference certain records that already exist in NetSuite, including at least one cost category and one item, as described below.

Cost Category

You reference an existing cost category record using the costCategory element. Note that the category referenced must use one of the cost types designed for use with manufacturing routing. These types are described in Defining Cost Categories for Routing. This topic also describes important restrictions regarding cost types that can be referenced only one time on a cost template record. Failure to follow these guidelines results in an error.

Item

You reference an existing item record using the item element. The function of the item record is to represent a specific expense that belongs to a manufacturing routing cost category. Only certain types of items can be set up this way. For more details, see the help topic Defining a Manufacturing Charge Item.

Sample Code

The following code illustrates how to add a manufacturing cost template record.

Java

```
public void testAddRecord() throws Exception

{

// This operation requires a valid session
this.login(true);

ManufacturingCostTemplate mct = new ManufacturingCostTemplate();

//subsidiary
mct.setSubsidiary(new RecordRef(null, "1", null, RecordType.subsidiary));

//name
mct.setName("ManufacturingCostTemplate");

//memo
mct.setMemo("ManufacturingCostTemplateMemo");

//cost detail
ManufacturingCostTemplateCostDetailList mctcdl = new ManufacturingCostTemplateCostDetailList();


ManufacturingCostTemplateCostDetail mctcd1 = new ManufacturingCostTemplateCostDetail();

//line #1
mctcd1.setCostCategory(new RecordRef(null, "1", null, RecordType.costCategory));
mctcd1.setItem(new RecordRef(null, "4639", null, RecordType.otherChargePurchaseItem));

mctcdl.setManufacturingCostTemplateCostDetail(new ManufacturingCostTemplateCostDetail[] {mctcd1
});

mct.setCostDetailList(mctcdl);

c.addRecord(mct);
}
```

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: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">nlbuild@ne
tsuite.com</ns2:email>
            <ns3:password xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">passwor
d</ns3:password>
            <ns4:account xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com">3604360<
/ns4:account>
            <ns5:role internalId="3" xmlns:ns5="urn:core_2017_1.platform.webservices.netsuite.c
om"/>
         </ns1:passport>
      </soapenv:Header>
   <soapenv:Body>
      <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
         <record xsi:type="ns6:ManufacturingCostTemplate" xmlns:ns6="urn:supplychain_2017_1.lis
ts.webservices.netsuite.com">
            <ns6:subsidiary internalId="1" type="subsidiary" xsi:type="ns7:RecordRef" xmlns:ns7
="urn:core_2017_1.platform.webservices.netsuite.com"/>
            <ns6:name xsi:type="xsd:string">ManufacturingCostTemplate</ns6:name>
            <ns6:memo xsi:type="xsd:string">ManufacturingCostTemplateMemo</ns6:memo>
            <ns6:costDetailList replaceAll="false" xsi:type="ns6:ManufacturingCostTemplateCostD
etailList">
               <ns6:manufacturingCostTemplateCostDetail xsi:type="ns6:ManufacturingCostTemplate
CostDetail">
                  <ns6:costCategory internalId="1" type="costCategory" xsi:type="ns8:RecordRef"
```

```
 xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com"/>
                 <ns6:item internalId="4639" type="otherChargePurchaseItem" xsi:type="ns9:Reco
rdRef" xmlns:ns9="urn:core_2017_1.platform.webservices.netsuite.com"/>
              </ns6:manufacturingCostTemplateCostDetail>
           </ns6:costDetailList>
        </record>
     </add>
  </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:Header>
        <platformMsgs:documentInfo xmlns:platformMsgs="urn:messages_2017_1.platform.webservice
s.netsuite.com">
        <platformMsgs:nsId>WEBSERVICES_3604360_031120131489538171800510632_a6b8a93ca56e</platf
ormMsgs:nsId>
        </platformMsgs:documentInfo>
     </soapenv:Header>
     <soapenv:Body>
        <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
           <writeResponse>
              <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
              <baseRef internalId="4402" type="manufacturingCostTemplate" xsi:type="platformCo
re:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
           </writeResponse>
        </addResponse>
     </soapenv:Body>
  </soapenv:Envelope>
```

Manufacturing Routing

If the Manufacturing Routing and Work Center feature has been enabled, you can use SuiteTalk to interact with manufacturing routing records. You can check to see whether the feature is enabled by going to Setup > Company > Enable Features, and reviewing the Items & Inventory tab.

The benefit of this feature is that it extends the Work In Process (WIP) feature, allowing you to specify a sequence of steps required for the completion of a WIP work order. You take advantage of this feature using a few records, including manufacturing routing. The routing record defines a series of tasks that must be completed by specific employee groups. When you save a WIP work order that references a particular manufacturing routing record, each step described in the routing record becomes a manufacturing operation task.

For details on manually creating a routing record, see the help topic Creating a Manufacturing Routing. For more information about the Manufacturing Routing and Work Center feature, see the help topic Manufacturing Routing.

The manufacturing routing record is defined in the listScm XSD.

Supported Operations

The following operations can be used with the manufacturing routing record.

add | addList | delete | deleteList | get |getDeleted |getList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's manufacturing routing reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Refer to the following sections for more details on working with manufacturing routing records.

Prerequisites for Creating a Record

When you create a routing record, certain records must already exist in NetSuite, as follows:

Assembly Item

Each routing record is created for use with a specific assembly item. You reference an assembly item using the Item element. For details on assembly items, see the help topic Assembly Items. For details on using SuiteTalk to interact with assembly items, see Assembly Item (BOM Item).

Cost Template and Work Center

For each routing record you create, at least one entry is required in the routingStep sublist. Each sublist entry must reference one of the each of the following:

- Cost Template — You reference an existing cost template record using the manufacturingCostTemplate element. For more on the cost template record in general, see the help topic Creating Manufacturing Cost Templates. For details on using SuiteTalk to interact with cost templates, see Manufacturing Cost Template.
- Work Center — You set a value for work center using the manufacturingWorkCenter element. A work center is a static employee group record that has been flagged as a Manufacturing Work Center. For more details, see the help topic Creating Manufacturing Work Centers or Groups.

Sample Code

The following code illustrates how to add a manufacturing routing record.

Java

```
public void testAddRecord() throws Exception {

// This operation requires a valid session
this.login(true);

ManufacturingRouting mfgRouting = new ManufacturingRouting();

//BODY FIELDS
mfgRouting.setItem(mrr("167")); //Assembly Item
mfgRouting.setName("ManufacturingRouting1");
mfgRouting.setMemo("ManufacturingRouting1Memo");
mfgRouting.setSubsidiary(new RecordRef(null, "1", null, RecordType.subsidiary));

RecordRefList rfList = new RecordRefList();
rfList.setRecordRef(new RecordRef[] {new RecordRef(null, "1", null, RecordType.location)});

mfgRouting.setLocationList(rfList);
mfgRouting.setIsDefault(true);

//SUBLIST
ManufacturingRoutingRoutingStepList stepsList = new ManufacturingRoutingRoutingStepList();
ManufacturingRoutingRoutingStep step1 = new ManufacturingRoutingRoutingStep();
step1.setOperationSequence(1L);
step1.setOperationName("mfgRoutingOpName1");
step1.setManufacturingWorkCenter(new RecordRef(null, "3513", null, RecordType.entityGroup));
step1.setManufacturingCostTemplate(new RecordRef(null, "4301", null, RecordType.manufacturingCo
stTemplate));
step1.setSetupTime(1.11D);
step1.setRunRate(2.22D);

stepsList.setManufacturingRoutingRoutingStep(new ManufacturingRoutingRoutingStep[] {step1});

mfgRouting.setRoutingStepList(stepsList);

c.addRecord(mfgRouting);
```

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: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">nlbuild@ne
tsuite.com</ns2:email>
```

```
            <ns3:password xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">passwor
d</ns3:password>
            <ns4:account xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com">3604360<
/ns4:account>
            <ns5:role internalId="3" xmlns:ns5="urn:core_2017_1.platform.webservices.netsuite.c
om"/>
         </ns1:passport>
      </soapenv:Header>
      <soapenv:Body>
         <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record xsi:type="ns6:ManufacturingRouting" xmlns:ns6="urn:supplychain_2017_1.lists
.webservices.netsuite.com">
               <ns6:subsidiary internalId="1" type="subsidiary" xsi:type="ns7:RecordRef" xmlns:
ns7="urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns6:item internalId="167" xsi:type="ns8:RecordRef" xmlns:ns8="urn:core_2017_1.p
latform.webservices.netsuite.com"/>
               <ns6:locationList xsi:type="ns9:RecordRefList" xmlns:ns9="urn:core_2017_1.platfo
rm.webservices.netsuite.com">
               <ns9:recordRef internalId="1" type="location" xsi:type="ns9:RecordRef"/>
                              </ns6:locationList>
               <ns6:name xsi:type="xsd:string">ManufacturingRouting1</ns6:name>
                  <ns6:memo xsi:type="xsd:string">ManufacturingRouting1Memo</ns6:memo>
               <ns6:isDefault xsi:type="xsd:boolean">true</ns6:isDefault>
               <ns6:routingStepList replaceAll="false" xsi:type="ns6:ManufacturingRoutingRoutin
gStepList">
                  <ns6:manufacturingRoutingRoutingStep xsi:type="ns6:ManufacturingRoutingRoutin
gStep">
                  <ns6:operationSequence xsi:type="xsd:long">1</ns6:operationSequence>
                  <ns6:operationName xsi:type="xsd:string">mfgRoutingOpName1</ns6:operationName
>
                  <ns6:manufacturingWorkCenter internalId="3513" type="entityGroup" xsi:type="n
s10:RecordRef" xmlns:ns10="urn:core_2017_1.platform.webservices.netsuite.com"/>
                  <ns6:manufacturingCostTemplate internalId="4301" type="manufacturingCostTempl
ate" xsi:type="ns11:RecordRef" xmlns:ns11="urn:core_2017_1.platform.webservices.netsuite.com"/>

                  <ns6:setupTime xsi:type="xsd:double">1.11</ns6:setupTime>
                  <ns6:runRate xsi:type="xsd:double">2.22</ns6:runRate>
               </ns6:manufacturingRoutingRoutingStep>
            </ns6:routingStepList>
         </record>
      </add>
   </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:Header>
         <platformMsgs:documentInfo xmlns:platformMsgs="urn:messages_2017_1.platform.webservice
s.netsuite.com">
         <platformMsgs:nsId>WEBSERVICES_3604360_031120131494382039617284600_ea972a8d42cc</platf
ormMsgs:nsId>
```

```
         </platformMsgs:documentInfo>
      </soapenv:Header>
      <soapenv:Body>
         <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <writeResponse>
               <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
            <baseRef internalId="2201" type="manufacturingRouting" xsi:type="platformCore:Recor
dRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
         </writeResponse>
      </addResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Nexus

A nexus is a tax jurisdiction, usually defined at the country level. This type of entity is available when the Advanced Taxes feature is enabled, to allow users to manage and calculate taxes for different jurisdictions within the same NetSuite account. For details, see the help topic Advanced Taxes.

The nexus record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the nexus record.

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s nexus reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Payroll Item

Payroll items store values for payroll transaction line items. Payroll items are used with payroll transactions generated by the Payroll feature, but these transactions are managed by NetSuite and are generally not used in integrations. You will most likely be using payroll items with the Paycheck Journal transaction that is designed for use in web services integrations with external payroll systems. For more information, see the help topic Using the Paycheck Journal Feature.

The payroll item record is defined in the lists employees XSD.

Supported Operations

The following operations can be used with payroll item records:

add | addList | delete | deleteList | get | getDeleted | getList | search | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's payroll item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Payroll Item Types

The following table lists supported payroll item types and their internal IDs:

Payroll Item Type	Internal ID
Deduction	16
Earning : Addition	6
Earning : Commission	7
Earning : Salary	2
Earning : Sick	3
Earning : Vacation	4
Earning : Wage	5
Employer : Contribution	17
Employer : Expense	19
Tax	8

The Tax item type can be either an employee tax or a company tax, depending on whether the value of the employeePaid field is True or False.

Some payroll item types have hidden fields:

- For Earning types, liabilityAccount and vendor fields are hidden.
- For Deduction type, expenseAccount field is hidden.

- For Tax type with employeePaid=True, expenseAccount field is hidden.

Code Samples

The following code adds an Earning:Addition type payroll item.

SOAP Request

```
/*
   Request
   <soapenv:Body>
     <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
       <record externalId="Pineapple10732612" xsi:type="ns6:PayrollItem" xmlns:ns6="urn:employ
ees_2017_1.lists.webservices.netsuite.com">
         <ns6:subsidiary internalId="1" type="subsidiary" xsi:type="ns7:RecordRef" xmlns:ns7="
urn:core_2017_1.platform.webservices.netsuite.com"/>
         <ns6:itemType internalId="6" type="payrollItem" xsi:type="ns8:RecordRef" xmlns:ns8="u
rn:core_2017_1.platform.webservices.netsuite.com"/>
         <ns6:name xsi:type="xsd:string">Re-Employment Service Fund</ns6:name>
         <ns6:expenseAccount internalId="67" type="account" xsi:type="ns9:RecordRef" xmlns:ns9
="urn:core_2017_1.platform.webservices.netsuite.com"/>
       </record>
      </add>
   </soapenv:Body>
   */
```

Java

```
public void addPayrollItem_EarningAddition() throws Exception
   {
      // This operation requires a valid session
      this.login(true);

      PayrollItem pi =  new PayrollItem();

      RecordRef rrSubsidiary = new RecordRef();
      rrSubsidiary.setType(RecordType.subsidiary);
      rrSubsidiary.setInternalId("1"); //Parent Company

      pi.setSubsidiary(rrSubsidiary);
      pi.setName("Re-Employment Service Fund");

      RecordRef rrItemType = new RecordRef();
      rrItemType.setType(RecordType.payrollItem);
      rrItemType.setInternalId("6"); //Earning:Addition

      pi.setItemType(rrItemType);

      RecordRef rrExpenseAccount = new RecordRef();
      rrExpenseAccount.setType(RecordType.account);
      rrExpenseAccount.setInternalId("67"); //Expense Account : Dues & Subscriptions

      pi.setExpenseAccount(rrExpenseAccount);
      _port.add(pi);

   }
```

The following code adds a deduction type payroll item.

SOAP Request

```
 /*
     *  <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
     *      <record externalId="Melon10133881" xsi:type="ns6:PayrollItem"  xmlns:ns6="urn:emplo
yees_2017_1.lists.webservices.netsuite.com">
               <ns6:subsidiary internalId="1" type="subsidiary" xsi:type="ns7:RecordRef" xmlns:
ns7="urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns6:itemType internalId="16" type="payrollItem" xsi:type="ns8:RecordRef" xmlns:
ns8="urn:core_2017_1.platform.webservices.netsuite.com"/>
               <ns6:name xsi:type="xsd:string">Meal Charges</ns6:name>
               <ns6:liabilityAccount internalId="27" type="account" xsi:type="ns9:RecordRef" xm
lns:ns9="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </record>
         </add>
     *
     */
```

Java

```
  public void addPayrollItem_Deduction() throws Exception
    {
     // This operation requires a valid session
      this.login(true);


      PayrollItem pi =  new PayrollItem();

      RecordRef rrSubsidiary = new RecordRef();
      rrSubsidiary.setType(RecordType.subsidiary);
      rrSubsidiary.setInternalId("1"); //Parent Company

      pi.setSubsidiary(rrSubsidiary);
      pi.setName("Meal Charges");

      RecordRef rrItemType = new RecordRef();
      rrItemType.setType(RecordType.payrollItem);
      rrItemType.setInternalId("16"); //Earning:Addition

      pi.setItemType(rrItemType);
```

```
        RecordRef rrLiabilityAccount = new RecordRef();
        rrLiabilityAccount.setType(RecordType.account);
        rrLiabilityAccount.setInternalId("27"); //Accrued expenses

        pi.setLiabilityAccount(rrLiabilityAccount);
        _port.add(pi);

    }
```

The following code gets values for an Earning:Salary type payroll item.

SOAP Request

```
 /*
        <get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
          <baseRef internalId="2" type="payrollItem" xsi:type="ns6:RecordRef" xmlns:ns6="urn:cor
e_2017_1.platform.webservices.netsuite.com"/>
        </get>
    */
```

Java

```
public void readPayrollItem() throws Exception
    {
        // This operation requires a valid session
        this.login(true);


        RecordRef rrPayrollItem =  new RecordRef();
        rrPayrollItem.setType(RecordType.payrollItem);
        rrPayrollItem.setInternalId("2"); // internalId of existing Payroll Item

        PayrollItem pi = (PayrollItem)_port.get(rrPayrollItem).getRecord();
    }
```

The following code updates a payroll item.

SOAP Request

```
 /*
     * <update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record internalId="2" xsi:type="ns6:PayrollItem" xmlns:ns6="urn:employees_2017_1.l
ists.webservices.netsuite.com">
               <ns6:name xsi:type="xsd:string">Meal Charges class1</ns6:name>
            </record>
          </update>
     */
```

Java

```
public void updatePayrollItem() throws Exception
    {
       // This operation requires a valid session
       this.login(true);


       PayrollItem pi = new PayrollItem();
       pi.setInternalId("2");
       pi.setName("Meal Charges class1");

       WriteResponse wr = _port.update(pi);

       if(wr.getStatus().isIsSuccess()){

          RecordRef rrPayrollItem =  new RecordRef();
          rrPayrollItem.setInternalId("2");
          rrPayrollItem.setType(RecordType.payrollItem);

          PayrollItem piGet = (PayrollItem)_port.get(rrPayrollItem).getRecord();
          System.out.println("PayrollItem Name :" + piGet.getName());
       }
}
```

The following code deletes a payroll item.

SOAP Request

```
   /*
      * <soapenv:Body>
          <delete xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
             <baseRef internalId="2" type="payrollItem" xsi:type="ns6:RecordRef" xmlns:ns6="urn:
core_2017_1.platform.webservices.netsuite.com"/>
          </delete>
       </soapenv:Body>
      */
```

Java

```
public void deletePayrollItem() throws Exception
    {
       // This operation requires a valid session
        this.login(true);


        RecordRef rrPayrollItem = new RecordRef();
        rrPayrollItem.setInternalId("2");
        rrPayrollItem.setType(RecordType.payrollItem);
        _port.delete(rrPayrollItem);
    }
```

The following code gets payroll items deleted on a specific date. In the SOAP request, the date is explicitly set. In the Java code, the date is determined by the getTodaysDate function.

SOAP Request

```
/*
     *  <soapenv:Body>
         <getDeleted xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <getDeletedFilter>
               <ns6:deletedDate operator="on" xmlns:ns6="urn:core_2017_1.platform.webservices.n
etsuite.com">
                  <ns6:searchValue>2012-07-24T16:00:00.000Z</ns6:searchValue>
               </ns6:deletedDate>
               <ns7:type operator="anyOf" xmlns:ns7="urn:core_2017_1.platform.webservices.netsu
ite.com">
                  <ns7:searchValue>payrollItem</ns7:searchValue>
               </ns7:type>
            </getDeletedFilter>
         </getDeleted>
       </soapenv:Body>
     *
     */
```

Java

```
public void getDeletedPayrollItem() throws Exception
   {
      // This operation requires a valid session
       this.login(true);

      GetDeletedFilter gdf =  new GetDeletedFilter();

      SearchDateField sdf =  new SearchDateField();
      sdf.setOperator(SearchDateFieldOperator.on);
      sdf.setSearchValue(Util.getTodaysDate());

      gdf.setDeletedDate(sdf);

      SearchEnumMultiSelectField sems =  new SearchEnumMultiSelectField();
      sems.setOperator(SearchEnumMultiSelectFieldOperator.anyOf);
      sems.setSearchValue(new String[] {"payrollItem"});
      gdf.setType(sems);

      _port.getDeleted(gdf);
   }
```

Revenue Recognition Schedule

A revenue recognition schedule indicates the posting periods in which revenue should be recognized, and the amount to be recognized in each period, for an item sale. A revenue recognition schedule is generated for any sales transaction item that has an associated revenue recognition template. The point at which a revenue recognition schedule is generated for an item sale depends upon the type of sales transaction and enabled features and preferences set in your account. The schedule could be generated when a transaction is first saved, when it is approved, or when it is billed. Revenue recognition schedules provide a basis for the generation of journal entries that record the impact of item sales. This record is available when the Revenue Recognition feature is enabled. For more details, see the help topic Working with Revenue Recognition Schedules.

Revenue recognition schedules are system-generated. In web services, you can get or search this type of record's data, and edit the name field only. To make other changes, you need to modify the associated revenue recognition template. See Revenue Recognition Template.

The Revenue recognition schedule record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the revenue recognition schedule record.

get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's revenue recognition schedule reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

If you are using an endpoint that predates 2014.2, be aware of the following: Some fields have been deprecated for both RevRecScheduleSearchBasic and RevRecScheduleSearchRowBasic. That is, regardless of which endpoint you are using, these fields are no longer honored.

The fields no longer honored in RevRecScheduleSearchBasic include the following:

- defRev
- entity
- externalId
- externalIdString

- incomeAcct
- srcDoc
- srcDocDate

The fields no longer honored in RevRecScheduleSearchRowBasic include the following:

- defRev
- entity
- externalId
- incomeAcct
- postPeriod
- srcDoc

Revenue Recognition Template

A revenue recognition template indicates how revenue from associated items should be posted. For each template: you can select from a choice of standard terms or define your own custom terms, set the time period over which recognition occurs, define an offset to delay the start of recognition, and set up an initial amount to be recognized. This record is available when the Revenue Recognition feature is enabled. For more information, see the help topic Creating Revenue Recognition Templates.

A revenue recognition schedule is generated for each sales transaction item that has an associated revenue recognition template is saved. See Payroll Item.

The revenue recognition template record is defined in the listAcct (accounting) XSD.

Note: To recognize revenue for the sale of an item, you must associate a deferred revenue account with that item on its item record. You can set a revenue recognition template on an item record; this template becomes the default for all sales of the item. You also can associate a revenue recognition template with an item on the item line of a transaction record, to apply only to that specific item sale. For more information, see the help topic Associating Revenue Recognition Templates with Items.

Supported Operations

The following operations can be used with the revenue recognition template record.

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s revenue recognition template reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Sales Tax Item

The sales tax item record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the sales tax item record.

add | addList | delete | deleteList | get | getAll | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s sales tax item reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Subsidiary

The subsidiary record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the subsidiary record.

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s subsidiary reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Note for Accounts Using the 2013.2 and Earlier WSDLs

If you are using the 2013.2 endpoint or earlier, be aware that the subsidiary field, as defined in your WSDL, includes some body fields that are no longer used by NetSuite. Therefore, even if you are using an older WSDL that lists these fields, you can no longer interact with them. The fields are:

- anonymousCustomerInboundEmail
- anonymousCustomerOnlineForms
- caseAssignmentTemplate
- caseAutomaticClosureTemplate
- caseCopyEmployeeTemplate
- caseCreationTemplate
- caseEscalationTemplate
- caseUpdateTemplate
- companyNameForSupportMessages
- employeeCaseUpdateTemplate
- mainSupportEmailAddress

Tax Group

The tax group record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the tax group record.

add | addList | delete | deleteList | get | getAll | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s tax group reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Tax Type

The tax type record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the tax type record.

add | addList | delete | deleteList | get | getAll | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's tax type reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Unit Type

The unit type record type is also known as Unit of Measure. This record type is available when the Multiple Units of Measure feature is enabled. For details, see the help topic Setting Up Units of Measure.

The unit type record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the unit type record.

add | addList | delete | deleteList | get | getAll | getList | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's units type reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Other Lists

For general information about other lists in SuiteTalk, see:

- Usage Notes for Other List Record Types

The following other list record types are supported in SuiteTalk:

- Budget Category
- Contact Category
- Contact Role
- Cost Category
- Customer Category
- Customer Message
- Global Account Mapping
- Item Account Mapping
- Lead Source
- Note Type
- Other Name Category
- Partner Category
- Payment Method
- Price Level
- Pricing Group
- Sales Role
- Tax Control Account
- Term
- Vendor Category
- Win Loss Reason

Usage Notes for Other List Record Types

Other list records represent user-defined lists that define items that can be selected for a specific field.

For example, in the contact record, you can set the type of contact in the *category* field. The *category* field references the ContactCategory record, which contains a predefined list of different contact types.

All other list record types are defined in the listAcct (accounting) XSD.

Permissions

In cases where the logged in user's role does NOT have permissions for the specified other list record, the get() operation will return only the key/name information instead of the whole record. This is to ensure that these lists can be manipulated when they are included as select or multiselect fields within another record.

Budget Category

Budget categories are used with the **Multiple Budgets** feature to create budgets for a variety of scenarios. Be aware that budget categories are available only if you have the Multiple Budgets feature enabled in your account. See Usage Notes for steps on enabling this feature.

The budget category record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the budget category record.

add | addList | delete | deleteList | get | getAll | getList | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the budget category reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

To use the budget category record, a NetSuite administrator must first enable the Multiple Budgets feature in your account.

To enable Multiple Budgets:

1. Go to Setup > Company > Enable Features.
2. Click the Accounting tab.
3. Select the Multiple Budget check box.
4. Click Save.

After this feature is enabled, any user can create budget categories in the UI. For steps on creating budget categories, see the help topic Creating Budget Categories for Local Subsidiary Budgeting in the NetSuite Help Center.

Contact Category

Contact category defines a list of values that are used by the contact record to set the type of contact. In the UI, this is a user defined list at Setup > Sales > CRM Lists > Contact Category. For more information, see the help topic Setting Up Accounting Lists.

Supported Operations

The following operations can be used with the contact category record.

add | addList | delete | deleteList | get | getList | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's contact category reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Contact Role

Contact roles can be defined for each contact listed on various records including opportunity, customer, vendor, and partner.

Supported Operations

The following operations can be used with the contact role record.

add | addList | delete | deleteList | get | getList | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For more details, see the Schema Browser's contact role reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Cost Category

Cost category records are used to classify different types of costs associated with your items. Using cost categories helps you to track costs and variances in the manufacturing process. These categories are available when the Standard Costing or Landed Cost feature is enabled.

Material or service cost categories track standard costs for items. For details, see the help topic Creating Cost Categories.

Landed cost category values are used for items on bills and item receipts to categorize the different kinds of expenses incurred when making purchases. For details, see the help topic Landed Cost Categories.

The cost category record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the cost category record.

add | addList | delete | deleteList | get | getAll | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For more details, see the Schema Browser's cost category reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Customer Category

Customer category defines a list of values that are used by the customer record to set the type of customer (see Customer). In the UI, reference an existing customer category list by going to Setup > Accounting > Accounting Lists.

To create a new customer category list, set the Type dropdown list at the top of the page to Customer Category. Next, click the New button that appears at the top of the page.

Supported Operations

The following operations can be used with the customer category record.

add | addList | delete | deleteList | get | getList | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's customer category reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Customer Message

Standardized customer messages can be created to be available for selection on transaction records. A selected message is sent to the customer for the transaction. For example, a message of "Thank you for your order!" can be selected on an invoice record to be displayed on the invoice sent to the customer. For information about how to define customer messages, see the help topic Setting Up Accounting Lists.

The customer message record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the customer message record.

add | addList | delete | deleteList | get | getList | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's customer message reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Global Account Mapping

For accounts using Multi-Book Accounting, the global account mapping record enables you to configure secondary accounting books to post to accounts different from the primary book. These mappings are used by transactions where the user can manually select the account to which the transaction posts.

This record is defined in the listAcct (Accounting) XSD, where it is called GlobalAccountMapping.

Using this record requires that, as part of your Multi-Book configuration, you select the Chart of Accounts Mapping option at Setup > Enable Features, on the Accounting subtab.

In the user interface, you access this record at Setup > Accounting > Global Account Mappings > New.

For more details on using SuiteTalk to work with the global account mapping record, see the following sections:

- Global Account Mapping Supported Operations
- Global Account Mapping Field Definitions
- Global Account Mapping Code Samples
- Common Errors With Global Account Mappings

Global Account Mapping Supported Operations

The following operations are supported for use with the global account mapping record:

add | addList | delete | deleteList | get | getDeleted | getList | search | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

For examples of how to use some of these operations with the global account mapping record, see Global Account Mapping Code Samples.

Global Account Mapping Field Definitions

SuiteTalk supports all the fields that are available in the user interface for the global account mapping record. It also supports external ID, any custom fields that you add, and the selection of a custom entry form (via the customForm field).

Note that the Custom Dimension field is available only if you have configured it at Setup > Accounting > Accounting Preferences, on the Accounting Books subtab. For details on this process, see the help topic Custom Mapping Dimensions.

For details about working with this record in the user interface, see the help topic Global Account Mapping.

Refer also to the SuiteTalk Schema Browser, which includes definitions for all body fields, sublist fields, search filters, and search joins available to this record.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Global Account Mapping Code Samples

For examples of how to work with global account mappings, refer to the following sections:

- Adding a Global Account Mapping Record

- Getting a Global Account Mapping Record

Adding a Global Account Mapping Record

The following example shows how to create a global account mapping record. This example assumes that the Custom Dimension field has been configured to reference currency type. Note also that this example does not set an end date, which means that the mapping will work indefinitely.

C#

```
private void addGlobalAccountMapping()
{

   // create object

   GlobalAccountMapping newGlobalAccountMapping = new GlobalAccountMapping();


   // create external Id

   newGlobalAccountMapping.externalId = "101A";


   // set effective date

   DateTime lastWeek = new DateTime(2013, 1, 20);
   newGlobalAccountMapping.effectiveDate = lastWeek;
   newGlobalAccountMapping.effectiveDateSpecified = true;


   // set accounting book

   RecordRef accountingBook = new RecordRef();
   accountingBook.internalId = "2";
   newGlobalAccountMapping.accountingBook = accountingBook;


   // set subsidiary

   RecordRef subsidiaryRef = new RecordRef();
   subsidiaryRef.internalId = "2";
   newGlobalAccountMapping.subsidiary = subsidiaryRef;


   // set source account

   RecordRef sourceAccount = new RecordRef();
   sourceAccount.internalId = "112";
   newGlobalAccountMapping.sourceAccount = sourceAccount;


   // set destination account

   RecordRef destinationAccount = new RecordRef();
   destinationAccount.internalId = "11";
   newGlobalAccountMapping.destinationAccount = destinationAccount;


   // set custom dimension - in this case, the custom dimension field
   // has been configured to reference a particular currency.

   RecordRef currencyType = new RecordRef();
   currencyType.internalId = "1";
   newGlobalAccountMapping.customDimension = currencyType;


   //execute add operation

   _service.add(newGlobalAccountMapping);

}
```

SOAP Request

```
<soap:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record xsi:type="q1:GlobalAccountMapping" xmlns:q1="urn:accounting_2017_1.lists.webservi
ces.netsuite.com">
         <q1:effectiveDate>2013-01-20T00:00:00</q1:effectiveDate>
         <q1:accountingBook internalId="2"/>
         <q1:subsidiary internalId="2"/>
       <q1:sourceAccount internalId="112"/>
       <q1:destinationAccount internalId="11"/>
         <q1:customDimension internalId="1" xsi:type="q2:RecordRef" xmlns:q2="urn:core_2017_1.p
latform.webservices.netsuite.com"/>
    </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="globalAccountMapping" internalId="2"
xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Getting a Global Account Mapping Record

The process of getting a global account mapping record is similar to the process of getting most other records.

However, you may notice one difference related to the custom dimension field. This difference occurs if you configure the custom dimension field to reference a custom record. In these cases, when you get a global account record that uses the custom dimension field, the SOAP response will include a CustomRecordRef, with an “internalId” referencing the custom record’s internal ID, and a “typeId” representing the custom record type’s internal ID. However, be aware that when you add a record, you can set a value for custom dimension using RecordRef, even when the field references a custom record.

The following example illustrates the scenario where custom dimension references a custom record.

C#

```
private void getGlobalAccountMapping()
{
   RecordRef recordGlobalAccountMapping = new RecordRef();
   recordGlobalAccountMapping.externalId = "101L";
   recordGlobalAccountMapping.type = RecordType.globalAccountMapping;
   recordGlobalAccountMapping.typeSpecified = true;

   _service.get(recordGlobalAccountMapping);
}
```

SOAP Request

```
<soap:Body>
   <get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <baseRef type="globalAccountMapping" externalId="101L" xsi:type="q1:RecordRef" xmlns:q1="
urn:core_2017_1.platform.webservices.netsuite.com"/>
   </get>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
   <getResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <readResponse>
         <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.web
services.netsuite.com"/>
            <record xsi:type="listAcct:GlobalAccountMapping" externalId="101L" internalId="901"
 xmlns:listAcct="urn:accounting_2017_1.lists.webservices.netsuite.com">
               <listAcct:customForm internalId="-720" xmlns:platformCore="urn:core_2017_1.platf
orm.webservices.netsuite.com">
                  <platformCore:name>Standard Global Account Mapping Form</platformCore:name>
               </listAcct:customForm>
               <listAcct:effectiveDate>2013-01-28T00:00:00.000-08:00</listAcct:effectiveDate>
               <listAcct:accountingBook internalId="8" xmlns:platformCore="urn:core_2017_1.plat
form.webservices.netsuite.com">
                  <platformCore:name>CAD Book all sub</platformCore:name>
               </listAcct:accountingBook>
               <listAcct:subsidiary internalId="2" xmlns:platformCore="urn:core_2017_1.platform
.webservices.netsuite.com">
                  <platformCore:name>CAN Subsid</platformCore:name>
               </listAcct:subsidiary>
               <listAcct:sourceAccount internalId="183" xmlns:platformCore="urn:core_2017_1.pla
tform.webservices.netsuite.com">
                  <platformCore:name>VAT on Sales</platformCore:name>
               </listAcct:sourceAccount>
               <listAcct:destinationAccount internalId="195" xmlns:platformCore="urn:core_2017_
1.platform.webservices.netsuite.com">
                   <platformCore:name>VAT on Sales DE</platformCore:name>
               </listAcct:destinationAccount>
               <listAcct:customDimension xsi:type="platformCore:CustomRecordRef" typeId="1" int
ernalId="2" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                  <platformCore:name>Phase</platformCore:name>
               </listAcct:customDimension>
            </record>
         </readResponse>
   </getResponse>
</soapenv:Body>
```

When adding this same record, you could do so using the following code snippet to identify the custom record:

```
RecordRef customRecord = new RecordRef();
customRecord.type = RecordType.customRecord;
customRecord.internalId = "2";
newGlobalAccountMapping.customDimension = customRecord;
```

Common Errors With Global Account Mappings

When you are trying to create global account mapping records, the following failure notices can occur in your SOAP responses:

- "A mapping rule with these dimensions already exists" — This error indicates that the mapping is identical to an existing mapping. You must change at least one of the mapping criteria, such as the source account or the custom dimension, before you can add the new mapping record.
- "Invalid destinationaccount reference key *x* for subsidiary *x*" — This error indicates that the destination account you chose is invalid, possibly because it is not the same account type as the source account. For more details on mapping rules, see the help topic Mapping Restrictions.

Item Account Mapping

For accounts using Multi-Book Accounting, the item account mapping record enables you to configure secondary accounting books to post to accounts different from the primary book, based on the item that is the subject of the transaction. These mappings are used by transactions where the item determines the account to which the transaction posts.

This record is defined in the listAcct (Accounting) XSD, where it is called ItemAccountMapping.

Using this record requires that, as part of your Multi-Book configuration, you select the Chart of Accounts Mapping option at Setup > Enable Features, on the Accounting subtab.

In the user interface, you access this record at Setup > Accounting > Item Account Mappings > New.

For more details on using SuiteTalk to work with the item account mapping record, see the following sections:

- Item Account Mapping Supported Operations
- Item Account Mapping Field Definitions
- Item Account Mapping Code Samples
- Common Errors With Item Account Mappings

Item Account Mapping Supported Operations

The following operations are supported for use with the item account mapping record:

add | addList | delete | deleteList | get | getDeleted | getList | search | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

For examples of how to use some of these operations with the item account mapping record, see Item Account Mapping Code Samples.

Item Account Mapping Field Definitions

SuiteTalk supports all the fields that are available in the user interface for the item account mapping record. It also supports external ID, any custom fields that you add, and the selection of a custom entry form (via the customForm field).

Note that the Custom Dimension field is available only if you have configured it at Setup > Accounting > Accounting Preferences, on the Accounting Books subtab. For details on this process, see the help topic Custom Mapping Dimensions.

For details about working with this record in the user interface, see the help topic Item Account Mapping. Refer also to the SuiteTalk Schema Browser, which includes definitions for all body fields, sublist fields, search filters, and search joins available to this record.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

For details about working with this record in the user interface, see the help topic Item Account Mapping.

Item Account Mapping Code Samples

For examples of how to work with item account mappings, refer to the following sections:

- Adding an Item Account Mapping Record
- Updating an Item Account Mapping Record

Adding an Item Account Mapping Record

The following example shows how to create an item account mapping record. In this example, the Custom Dimension field has been configured to reference billing schedule. The request also specifies the use of a custom entry form.

C#

```
private void addItemAccountMapping()
{

    // create object

      ItemAccountMapping newItemAccountMapping = new ItemAccountMapping();


      // specify the custom entry form that should be used

      RecordRef myCustomForm = new RecordRef();
      myCustomForm.type = RecordType.customRecord;
      myCustomForm.typeSpecified = true;
      myCustomForm.internalId = "102";
      newItemAccountMapping.customForm = myCustomForm;


      // create external Id

      newItemAccountMapping.externalId = "106D";


      // set effective date

      DateTime lastWeek = new DateTime(2013, 1, 28);
      newItemAccountMapping.effectiveDate = lastWeek;
      newItemAccountMapping.effectiveDateSpecified = true;


      // set end date

      DateTime nextYear = new DateTime(2014, 1, 28);
      newItemAccountMapping.endDate = nextYear;
      newItemAccountMapping.endDateSpecified = true;


      // set accounting book

      RecordRef accountingBook = new RecordRef();
      accountingBook.internalId = "2";
      newItemAccountMapping.accountingBook = accountingBook;


      // set subsidiary

      RecordRef subsidiaryRef = new RecordRef();
      subsidiaryRef.internalId = "2";
      newItemAccountMapping.subsidiary = subsidiaryRef;


      // set item account type; this choice determines
      // the exact accounts available when you select
      // source account and destination account.

      newItemAccountMapping.itemAccount = ItemAccountMappingItemAccount._asset;
      newItemAccountMapping.itemAccountSpecified = true;


      // set source account

      RecordRef mySourceAccount = new RecordRef();
      mySourceAccount.type = RecordType.account;
      mySourceAccount.typeSpecified = true;
      mySourceAccount.internalId = "9";
      newItemAccountMapping.sourceAccount = mySourceAccount;


      // set destination account

      RecordRef myDestinationAccount = new RecordRef();
      myDestinationAccount.type = RecordType.account;
      myDestinationAccount.typeSpecified = true;
      myDestinationAccount.internalId = "11";
      newItemAccountMapping.destinationAccount = myDestinationAccount;


      // set custom dimension - in this case, the custom dimension field
      // has been configured to reference billing schedule.

      RecordRef myBillingSchedule = new RecordRef();
      myBillingSchedule.type = RecordType.billingSchedule;
      myBillingSchedule.typeSpecified = true;
      myBillingSchedule.internalId = "6";
      newItemAccountMapping.customDimension = myBillingSchedule;


      // execute add operation

      _service.add(newItemAccountMapping);

}
```

SOAP Request

```
<soap:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record externalId="106D" xsi:type="q1:ItemAccountMapping" xmlns:q1="urn:accounting_2017_
1.lists.webservices.netsuite.com">
         <q1:customForm type="customRecordType" internalId="102"/>
         <q1:effectiveDate>2013-01-28T00:00:00</q1:effectiveDate>
         <q1:endDate>2014-01-28T00:00:00</q1:endDate>
         <q1:accountingBook internalId="2"/>
         <q1:subsidiary internalId="2"/>
         <q1:itemAccount>_asset</q1:itemAccount>
         <q1:sourceAccount type="account" internalId="9"/>
         <q1:destinationAccount type="account" internalId="11"/>
         <q1:customDimension type="billingSchedule" internalId="6" xsi:type="q2:RecordRef" xmln
s:q2="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </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="itemAccountMapping" externalId="106D"
 internalId="404" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Updating an Item Account Mapping Record

The following example shows how to update an item account mapping record. In this example, the update specifies the use of a different entry form, changes the destination account, and sets the Custom Dimension field to a value of “unassigned.” Be aware that the Custom Dimension field is available only if you have configured it, at Setup > Accounting > Accounting Preferences, on the Accounting Books subtab.

C#

```
private void updateItemAccountMapping()
{

   // create object

   ItemAccountMapping updatedItemAccountMapping = new ItemAccountMapping();


   // identify the record being updated

   updatedItemAccountMapping.externalId = "106D";


   // change the form that is being used

   RecordRef myCustomForm = new RecordRef();
   myCustomForm.type = RecordType.customRecord;
   myCustomForm.typeSpecified = true;
   myCustomForm.internalId = "-730";
   updatedItemAccountMapping.customForm = myCustomForm;


   // change the destination account

   RecordRef myDestinationAccount = new RecordRef();
   myDestinationAccount.type = RecordType.account;
   myDestinationAccount.typeSpecified = true;
   myDestinationAccount.internalId = "175";
   updatedItemAccountMapping.destinationAccount = myDestinationAccount;


   // change the custom dimension - update it to the status of "unassigned"
   // using internal ID -999

   RecordRef myBillingSchedule = new RecordRef();
   myBillingSchedule.type = RecordType.billingSchedule;
   myBillingSchedule.typeSpecified = true;
   myBillingSchedule.internalId = "-999";
   updatedItemAccountMapping.customDimension = myBillingSchedule;


   // execute update operation

   _service.update(updatedItemAccountMapping);

}
```

SOAP Request

```
<soap:Body>
   <update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record externalId="106D" xsi:type="q1:ItemAccountMapping" xmlns:q1="urn:accounting_2017_
1.lists.webservices.netsuite.com">
         <q1:customForm type="customRecordType" internalId="-730"/>
         <q1:destinationAccount type="account" internalId="175"/>
         <q1:customDimension type="billingSchedule" internalId="-999" xsi:type="q2:RecordRef" x
mlns:q2="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </record>
   </update>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
   <updateResponse 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="itemAccountMapping" externalId="106D"
 internalId="404" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
```

```
        </writeResponse>
    </updateResponse>
</soapenv:Body>
```

Common Errors With Item Account Mappings

When you are trying to create item account mapping records, the following failure notices can occur in your SOAP responses:

- "A mapping rule with these dimensions already exists" — This error indicates that the mapping is identical to an existing mapping. You must change at least one of the mapping criteria, such as the source account or the custom dimension, before you can add the new mapping record.
- "Invalid destinationaccount reference key *x* for subsidiary *x*" — This error indicates that the destination account you chose is invalid, possibly because it is not the same account type as the source account. For more details on mapping rules, see the help topic Mapping Restrictions.

Lead Source

LeadSource defines a list of values that are used by the customer record to set the source of the lead for this customer — such as an ad or website referral (see Customer).

If you use the Marketing Automation feature, the list of possible lead sources matches the titles of your marketing campaigns. Administrators can enable this feature at Setup > Company > Enable Features > CRM.

If you do not use the Marketing Automation feature, in the UI, a user-defined lead source list is defined at Setup > Sales > Setup Tasks > CRM Lists > New > Lead Source. Note the Marketing Automation feature must be turned off before you can create a user-defined lead source list.

Supported Operations

The following operations can be used with the lead source record.

add | addList | delete | deleteList | getAll | getList | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Note: The getAll operation should be used to retrieve values of a list since the search operation does NOT exist for this record.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's lead source reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Note Type

This type defines a list of values that are used by the note record to set the type of note (see Note). In the UI, this is a user defined list at Setup > Sales > Setup Tasks > CRM Lists > New > Note Type.

Supported Operations

The following operations can be used with the note type record.

add | addList | delete | deleteList | get | getList | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's note type reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Other Name Category

Other name category values are used on other name records to categorize them. The list of other name records is a collection of records for people or companies who are not vendors, customers, or employees. You can use other name records to track other people or companies to whom you write checks or from whom you receive deposits.

For example, your company might donate money to a favorite charity, so you can create an other name record for the charity. You might also list your owners and partners here if they contribute or withdraw equity.

The other name category record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the other name category record.

add | addList | delete | deleteList | get | getList | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's other name category reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Partner Category

The partner category record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the partner category record.

add | addList | delete | deleteList | get | getList | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's partner category reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Payment Method

Supported Operations

The following operations can be used with the payment method record.

add | addList | delete | deleteList | get | getList | getSelectValue | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's payment method reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Price Level

Price level defines a list of values that are used by the opportunity and item records to set the price level for a specific item. Items can be assigned different price levels — such as Employee Price or Corporate Discount Price (see OpportunityItemList).

When the Multiple Pricing feature is enabled, you can create up to 1000 price levels for items. Setting up multiple price levels allows greater flexibility to set different pricing for different customers. For information, see the help topics Creating Price Levels and Setting Up Items for Multiple Price Levels.

Supported Operations

The following operations can be used with the price level record.

add | addList | delete | deleteList | get | getList | getSavedSearch | getSelectValue | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's price level reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Pricing Group

Pricing groups allow you to assign customer-specific price levels for groups of items. For more information, see the help topic Pricing Groups.

Supported Operations

The following operations can be used with the pricing group record.

add | addList | delete | deleteList | get | getList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's pricing group reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Sales Role

Sales roles can be defined for each member of a sales team. To use this record the Team Selling feature must be enabled. To enable team selling, go to Setup > Company > Enable Features > CRM. Click the Team Selling checkbox, and then click Save.

Supported Operations

The following operations can be used with the sales role record.

add | addList | delete | deleteList | getList | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's sales role reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Tax Control Account

A tax control account is an account to which the amounts computed for indirect taxes, such as sales tax and VAT, are posted.

The tax control account is defined in the listAcct (accounting) XSD, where it is called *taxAcct*.

In the UI, you can create a tax control account, and view existing ones, at Setup > Accounting > Tax Control Accounts. Note also that a tax control account is essentially an account record, with a few differences. Because they are accounts, tax control accounts also show up in the full list of accounts at Lists > Accounting > Accounts.

Supported Operations

The following operations can be used with the tax control account record.

add | addList | get | getList | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Updating

The update operation is permitted, but you can update only certain fields: Description, External ID, IsInactive, and Name. However, you may be able to change other fields when interacting with the tax control account as an account. For details, see Account.

Alternatives to Deleting

The delete operation is not permitted, because a tax control account cannot be deleted (either through the UI or through web services). If the Advanced Taxes feature is enabled, you can effectively remove the tax control account by deleting the nexus that the account is associated with. If the Advanced Taxes feature is *not* enabled, you cannot remove the account.

An alternative to deleting the account is to make it account inactive, by setting the IsInactive field to true.

For more details on accounts that cannot be deleted in NetSuite, refer to the help topic Deleting Accounts and Making Accounts Inactive.

Field Definitions

This section describes some of the key fields on the tax control account.

Country

The value for country is derived from the nexus value and is read-only. Note that, if your account does not have the Advanced Taxes feature enabled, the value for country will always be the same for all your tax control accounts.

Description

The description of the record can include only up to 50 characters. Otherwise, the operation fails with an error reading “The field description contained more than the maximum number (50) of characters allowed.” Description is one of the few fields that can be modified during an update.

Description is one of the few fields that can be modified during an update.

External ID

ExternalID is one of the few fields that can be modified during an update.

IsInactive

Because a tax control account cannot be deleted, you might want to make it inactive.

IsInactive is one of the few fields that can be modified during an update.

Name

The name of the record must be unique. If you try to add a tax control account using a non-unique value for the name field, the system returns an error reading “This record already exists,” even if you included a unique external ID.

Name is one of the few fields that can be modified during an update.

Nexus

You can set a value for nexus only if the Advanced Taxes feaure is enabled. If your NetSuite account does not use the Advanced Taxes feature, the value for nexus is set automatically and will always be the same for all your tax control accounts.

When Advanced Taxes is enabled, referencing a nexus value is required. However, after the account is created, the value for nexus cannot be changed.

Note that Advanced Taxes is enabled in all OneWorld accounts and cannot be turned off.

Tax Account Type

All tax control accounts have a tax account type, but you only actively choose a tax account type for accounts in certain countries. For example:

- When you are creating a tax code for an account in the United Kingdom, valid choices include _sale and _purchase. These two choices are elements of type accountingTypes: TaxAcctType.
- When creating a tax code for an account in the United States, only one option exists (_sale), so you do not set a value for this field.

In countries where both _sale and _purchase are valid choices, the tax account type field is required. Further, after an account has been created, the tax account type cannot be changed.

In countries where only one choice is allowed, tax account type is automatically set, and you cannot change it through web services (or through the UI).

Note that if your account uses Advanced Taxes and has nexuses in many countries, you may need to set this field for some tax control accounts and not others.

For More Information

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s tax account reference page.

Usage Notes

This section includes additional details on interacting with the tax record.

Finding the Internal ID

If you need the internal ID for an existing tax control account, note that you cannot find it through Setup > Accounting > Tax Control Accounts. However, you can find it at Lists > Accounting > Accounts. Make sure you have configured your NetSuite preference to "Show Internal IDs." (You can find this choice at Home > Set Preferences.)

When sorting accounts at Lists > Accounting > Accounts, be aware that:

- Tax control accounts of type "_sale" show up as Other Current Liability.
- Tax control accounts of type "_purchase" show up as Other Curret Asset.

Be aware that you can also add and update values for the external ID field.

Relationship to the Account Record

You can interact with the tax control account in many of the same ways you can interact with any account. However, some exceptions exist. For example, a tax control account cannot be deleted, and there are limits to what you can update, as described in Updating. However, when you interact with the record as an account, you may be able to update additional fields.

For details on interacting with the account record, see Account.

Code Samples

Refer to the following examples for help interacting with the tax control account record.

Example 1 — Creating a Record

The following example shows how to create a tax control account when the Advanced Taxes feature is enabled.

Java

```
public void testAddTaxAccount() throws Exception
   {
       c.useRequestLevelCredentials();

       TaxAcct taxAcct = new TaxAcct();

       RecordRef nexusRef = new RecordRef();
       nexusRef.setType(RecordType.nexus);
       nexusRef.setInternalId("6");

       taxAcct.setNexus(nexusRef);

       taxAcct.setName("WS Added TaxAcct");
       taxAcct.setDescription("WS Added Testing Tax Control Account");
       taxAcct.setTaxAcctType(TaxAcctType._purchase);
       taxAcct.setExternalId("WSTaxAcct1234567890");
```

```
        c.addRecord(taxAcct);
    }
```

SOAP Request

```
<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>
      <ns1:passport soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next" soapenv:mustUnde
rstand="0" xmlns:ns1="urn:messages_2017_1.platform.webservices.netsuite.com">
         <ns2:email xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">johnsmith@net
suite.com</ns2:email>
         <ns3:password xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">welcome</n
s3:password>
         <ns4:account xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com">1326288</ns
4:account>
         <ns5:role internalId="3" xmlns:ns5="urn:core_2017_1.platform.webservices.netsuite.com"
/>
      </ns1:passport>
   </soapenv:Header>
   <soapenv:Body>
      <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
         <record externalId="WSTaxAcct1234567890" xsi:type="ns6:TaxAcct" xmlns:ns6="urn:account
ing_2017_1.lists.webservices.netsuite.com">
            <ns6:name xsi:type="xsd:string">WS Added TaxAcct</ns6:name>
            <ns6:description xsi:type="xsd:string">WS Added Testing Tax Control Account</ns6:de
scription>
            <ns6:nexus internalId="6" type="nexus" xsi:type="ns7:RecordRef" xmlns:ns7="urn:core
_2017_1.platform.webservices.netsuite.com"/>
            <ns6:taxAcctType xsi:type="ns8:TaxAcctType" xmlns:ns8="urn:types.accounting_2017_1.
lists.webservices.netsuite.com">_purchase</ns6:taxAcctType>
         </record>
      </add>
   </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```
<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.n
etsuite.com">
         <platformMsgs:nsId>WEBSERVICES_1326288_042520131100061755935117479_6033b31321c2</platf
ormMsgs:nsId>
      </platformMsgs:documentInfo>
   </soapenv:Header>
   <soapenv:Body>
      <addResponse 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="286" externalId="WSTaxAcct1234567890" type="taxAcct" xsi:type=
"platformCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"
/>
         </writeResponse>
      </addResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Example 2 — Updating a Record

The following example shows how to modify a tax control account. Note that the only fields which can be updated are Name, Description, and IsInactive.

Java

```
 public void testUpdateTaxAccount() throws Exception
{
    c.useRequestLevelCredentials();
    Record taxAcctRec = c.getRecord("186", RecordType.taxAcct);
    TaxAcct taxAcct = (TaxAcct) taxAcctRec;

    taxAcct.setName("WS Updated TaxAcct nr 4");
    taxAcct.setDescription("WS Updated Testing Tax Control Account number 4");
    taxAcct.setExternalId("WSTaxAcct004");
    taxAcct.setIsInactive(true);

    c.updateRecord(taxAcct);
}
```

SOAP Request

```
<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>
      <ns1:passport soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next" soapenv:mustUnde
rstand="0" xmlns:ns1="urn:messages_2017_1.platform.webservices.netsuite.com">
         <ns2:email xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">johnsmith@net
suite.com</ns2:email>
         <ns3:password xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">welcome</n
s3:password>
         <ns4:account xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com">1326288</ns
4:account>
         <ns5:role internalId="3" xmlns:ns5="urn:core_2017_1.platform.webservices.netsuite.com"
/>
      </ns1:passport>
   </soapenv:Header>
   <soapenv:Body>
      <update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
         <record externalId="WSTaxAcct004" internalId="186" xsi:type="ns6:TaxAcct" xmlns:ns6="u
rn:accounting_2017_1.lists.webservices.netsuite.com">
            <ns6:name xsi:type="xsd:string">WS Updated TaxAcct nr 4</ns6:name>
            <ns6:description xsi:type="xsd:string">WS Updated Testing Tax Control Account numbe
```

```
r 4</ns6:description>
            <ns6:nexus internalId="1" xsi:type="ns7:RecordRef" xmlns:ns7="urn:core_2017_1.platf
orm.webservices.netsuite.com"/>
            <ns6:country xsi:type="ns8:Country" xmlns:ns8="urn:types.common_2017_1.platform.web
services.netsuite.com">_unitedStates</ns6:country>
            <ns6:taxAcctType xsi:type="ns9:TaxAcctType" xmlns:ns9="urn:types.accounting_2017_1.
lists.webservices.netsuite.com">_sale</ns6:taxAcctType>
            <ns6:isInactive xsi:type="xsd:boolean">true</ns6:isInactive>
         </record>
      </update>
   </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```
<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.n
etsuite.com">
         <platformMsgs:nsId>WEBSERVICES_1326288_042520131118845977808995829_ea95975e7048</platf
ormMsgs:nsId>
      </platformMsgs:documentInfo>
   </soapenv:Header>
   <soapenv:Body>
      <updateResponse 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="186" externalId="WSTaxAcct004" type="taxAcct" xsi:type="platfo
rmCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
         </writeResponse>
      </updateResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Example 3 — Getting a Record

The following example shows how to get a tax control account record.

Java

```
//Getting existing Tax Control Account

    public void testGetTaxAccount() throws Exception
    {
        c.useRequestLevelCredentials();
        RecordRef acct = new RecordRef();
   acct.setType(RecordType.taxAcct);
   acct.setExternalId("nove");
   c.getPort().get(acct);
    }
```

SOAP Request

```
<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>
        <ns1:passport soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next" soapenv:mustUn
derstand="0" xmlns:ns1="urn:messages_2017_1.platform.webservices.netsuite.com">
           <ns2:email xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">nlbuild@net
suite.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">3604360</
ns4:account>
           <ns5:role internalId="3" xmlns:ns5="urn:core_2017_1.platform.webservices.netsuite.co
m"/>
        </ns1:passport>
     </soapenv:Header>
     <soapenv:Body>
        <get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
           <baseRef externalId="nove" type="taxAcct" xsi:type="ns6:RecordRef" xmlns:ns6="urn:co
re_2017_1.platform.webservices.netsuite.com"/>
        </get>
     </soapenv:Body>
  </soapenv:Envelope
```

SOAP Response

```
<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.n
etsuite.com">
         <platformMsgs:nsId>WEBSERVICES_1326288_0425201311188459778089958 29_ea95975e7048</platf
ormMsgs:nsId>
      </platformMsgs:documentInfo>
   </soapenv:Header>
   <soapenv:Body>
      <updateResponse 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="186" externalId="WSTaxAcct004" type="taxAcct" xsi:type="platfo
rmCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
         </writeResponse>
      </updateResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Term

Terms are used to specify when payment is due on your customers' invoices. Define the specific requirements of a term of payment by creating a term record. You can create different payment terms for different customers. In the UI, this is a user defined list at Setup > Accounting > Setup Tasks > Accounting Lists > Term > New. For more information about payment terms, see the help topic Creating Terms of Payment.

Supported Operations

The following operations can be used with the term record.

add | addList | delete | deleteList | getDeleted | getList | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's term reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Vendor Category

The vendor category record is defined in the listAcct (accounting) XSD.

Supported Operations

The following operations can be used with the vendor category record.

add | addList | delete | deleteList | get | getList | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's vendor reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Sample Code

The following example shows how to add a vendor category through web services.

SOAP Request

```
      <soapenv:Body>
         <add xmlns="urn:messages_2017_2.platform.webservices.netsuite.com">
            <record xsi:type="ns7:VendorCategory" xmlns:ns7="urn:accounting_2017_2.lists.webser
vices.netsuite.com">
               <ns7:name xsi:type="xsd:string">Insurance Providers</ns7:name>
               <ns7:isTaxAgency xsi:type="xsd:boolean">false</ns7:isTaxAgency>
               <ns7:isInactive xsi:type="xsd:boolean">false</ns7:isInactive>
            </record>
         </add>
      </soapenv:Body>
```

Java

```
    public void vendorCategory() throws Exception {
        VendorCategory vendorCat = new VendorCategory();
        vendorCat.setName("Insurance Providers");
        vendorCat.setIsInactive(false);
        vendorCat.setIsTaxAgency(false);

        c.addRecord(vendorCat);
    }
```

Win Loss Reason

WinLossReason defines a list of values that are used by the opportunity record to set the reason for a win or loss (see Opportunity). In the UI, this is a user defined list at Setup > Sales > CRM Lists > New > Win/Loss Reason.

You cannot set a win loss reason for any transaction other than an opportunity.

Supported Operations

The following operations can be used with the win loss reason record.

add | addList | delete | deleteList | getDeleted | getList | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's win loss reasons reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Customization

The following customization records are supported in SuiteTalk:

- Custom Record
- Custom Record Type
- Custom Record Custom Field
- Custom List
- Custom Transaction
- CRM Custom Field
- Entity Custom Field
- Item Custom Field
- Item Number Custom Field
- Item Option Custom Field
- Other Custom Field
- Transaction Body Custom Field
- Transaction Column Custom Field

Note: If you are unfamiliar with NetSuite custom fields, it is recommended that you see the help topic Custom Fields in the NetSuite Help Center. This section describes the purpose and general characteristics of each custom field type, which will help you when working with the SuiteTalk customization API. For general information on working with both standard and custom fields in web services, see the help topic Fields in Web Services.

Custom Record

Custom records are entry forms based on existing record types, but customized to include fields for gathering information specific to the needs of your business. Use the custom record object to define records to emulate existing custom records.

The custom record object is defined in the setupCustom (customization) XSD.

Supported Operations

The following operations can be used with the custom record record type.

add | addList | delete | deleteList | getDeleted | get | getList | getSavedSearch | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's custom record reference page.

Note: If you are using the 2013.2 WSDL or later, you may notice that this record's Translations sublist is exposed. However, this sublist is available only in certain cases. Specifically, in order for the Translations sublist to be available, the corresponding record type must have the enableNameTranslation field set to a value of True. For more details on configuring a record type so that it allows for translations, see the help topic Translating Custom Record Instance Names. For details on working with custom record types via web services, see Custom Record Type.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Currently NetSuite does not support the **store values** option on custom records. Additionally, see the following:

- Using CustomRecordRef
- Understanding altName and autoName

Using CustomRecordRef

Further, note that when referencing custom records, you may need a separate CustomRecordRef type and a RecordRef type.

RecordRef and CustomRecordRef both descend from BaseRef. They are most prominently used in **get()**, and that is why they have a type attribute. CustomRecordRef has a typeId to indicate which **kind** of CustomRecord it is. Therefore, you cannot get a CustomRecord by setting the RecordRef.type to customRecord on add — you must use CustomRecordRef to specify the kind of CustomRecord.

Note: For a list of the possible typeIds (or record internalIds), refer to ListOrRecordRef in the NetSuite Help Center.

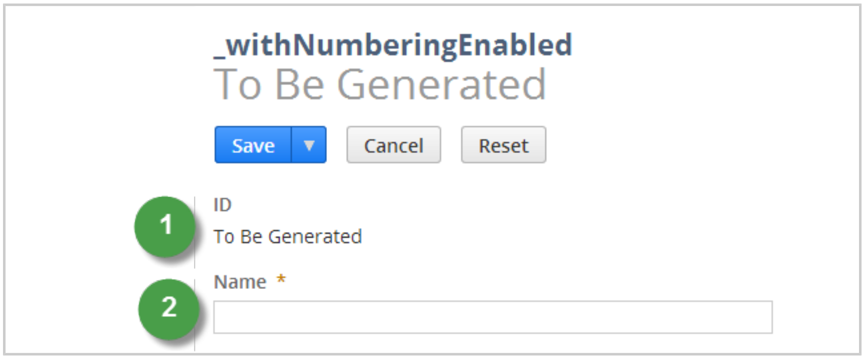
CustomRecord descends from Record, like Customer or Opportunity. This is what you submit to **add()** and it is what is returned through search(). CustomRecord uses a RecordRef() to store its type information.

Understanding altName and autoName

The altName and autoName fields are available only when you are creating a custom record based on a custom record type configured to use auto-generated numbering. (To check whether a custom record type is configured this way, look at the Enable box on the custom record type's Numbering tab.)

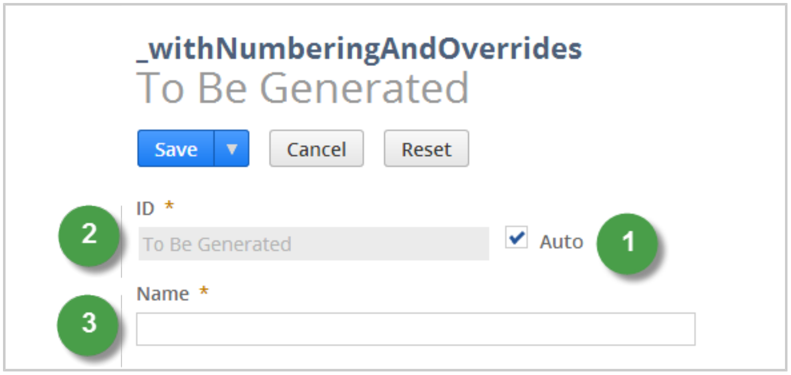
When you use this sort of custom record type as the basis for a new custom record, the altName field is required on the new custom record form. altName is a string field that represents a label you give the custom record, to be used in the list view. Be aware that the altName field maps to the field labeled Name in the user interface (which is different from the way the field names map when auto-generated numbers are not being used).

In this situation, the web services field called name maps to the field labeled ID in the user interface. This field represents the auto-generated string assigned to the custom record.



Callout	Description
1	The ID field is labeled **name** in web services.
2	The Name field is labeled **altName** in web services.

In some cases, a custom record type may be set up with both the Enable and Allow Override boxes selected. In these cases, you have the additional option of using the boolean autoName field. The autoName field maps to the box labeled Auto in the user interface. This field lets you disable auto-generated numbering for your new custom record. If you set autoName to false, you must provide a value for the name field (which, again, in the user interface is labeled ID). Note that in this situation, altName is also still required.



Callout	Description
1	The Auto box is labeled **autoName** in web services.
2	The ID field is labeled **name** in web services. This field is read only if the autoName value is true.
3	The Name field is labeled **altName** in web services.

Code Samples

SOAP

Request to get a custom record:

```
<soapenv:Body>
```

```
  <get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
        <baseRef internalId="102" typeId="139" xsi:type="ns6:CustomRecordRef" xmlns:ns6="urn:co
re_2017_1.platform.webservices.netsuite.com"/>
  </get>
</soapenv:Body>
```

SOAP response:

```
     <soapenv:Body>
         <getResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <readResponse>
               <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
               <record internalId="102" xsi:type="setupCustom:CustomRecord" xmlns:setupCustom="
urn:customization_2017_1.setup.webservices.netsuite.com">
                  <setupCustom:isInactive>false</setupCustom:isInactive>
                  <setupCustom:name>Truman Capote</setupCustom:name>
                  <setupCustom:owner internalId="79" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com">
                     <platformCore:name>Tom Smith</platformCore:name>
                  </setupCustom:owner>
                  <setupCustom:recType internalId="139" xmlns:platformCore="urn:core_2017_1.pla
tform.webservices.netsuite.com">
                     <platformCore:name>Author page</platformCore:name>
                  </setupCustom:recType>
                  <setupCustom:customFieldList xmlns:platformCore="urn:core_2017_1.platform.web
services.netsuite.com">
                     <platformCore:customField internalId="424" scriptId="custrecord_award_winn
er" xsi:type="platformCore:BooleanCustomFieldRef">
                        <platformCore:value>false</platformCore:value>
                     </platformCore:customField>
                  </setupCustom:customFieldList>
               </record>
            </readResponse>
         </getResponse>
      </soapenv:Body>
```

Request to add a custom record:

```
     <soapenv:Body>
         <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record xsi:type="ns6:CustomRecord" xmlns:ns6="urn:customization_2017_1.setup.webse
rvices.netsuite.com">
               <ns6:name xsi:type="xsd:string">Shirley Jackson</ns6:name>
               <ns6:recType internalId="139" xsi:type="ns7:RecordRef" xmlns:ns7="urn:core_2017_
1.platform.webservices.netsuite.com">
                  <ns7:name xsi:type="xsd:string">Author page</ns7:name>
               </ns6:recType>
            </record>
         </add>
      </soapenv: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.platfo
rm.webservices.netsuite.com"/>
               <baseRef internalId="202" typeId="139" xsi:type="platformCore:CustomRecordRef" x
mlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </writeResponse>
         </addResponse>
      </soapenv:Body>
```

Java

To add a custom record:

```
   CustomRecord myCR = new CustomRecord();
   RecordRef rt = new RecordRef();
   rt.setName("Authors");
   rt.setId("3"); // This indicates a typeId of 3 and corresponds to the Authors CustomRecord t
ype
   myCR.setRecType(rt);
   myCR.setName("Ernest Hemmingway"); // This is what will show up in List->
   CustomRecords->Authors under Name
   myCR.setCustomFieldList(myCFL); // An already filled out CustomField List
   WriteResponse wr = port.add(myCR);
```

Note that setTypeId is not submitted since this is the internal ID returned by the response, like for normal records.

To get a custom record:

```
CustomRecordRef customRec = new CustomRecordRef();
customRec.setTypeId("21"); // internal id of the custom record type
customRec.setInternalId("1"); // internal id of the custom record entry
sessMgr.getWrappedPort().get(customRec, this);
```

The following code retrieves the entire custom list for a list with an internal ID of **1**.

```
CustomRecordSearch customRecordSearch = new CustomRecordSearch();
RecordRef recordRef = new RecordRef();
recordRef.setInternalId("1");
customRecordSearch.setRecType(recordRef);
SearchResult result = port.search(customRecordSearch);
```

C#

To add a custom record:

```
   CustomRecord myCR = new CustomRecord();
   RecordRef rt = new RecordRef();
   rt.name = "Authors";
   // This indicates a typeId of 3 and corresponds to the Authors CustomRecord type
   rt.internalId = "3";
   customRecord.recType = rt;
```

```
// This is what will show up in List->CustomRecords->Authors under Name
customRecord.name = "Ernest Hemingway";
customRecord.customFieldList = myCFL; // An already filled out CustomField List
WriteResponse wr = port.add(customRecord);
```

To get a custom record:

```
CustomRecordRef myCRR = new CustomRecordRef();
myCRR.internalId = "1200"; // The key we got back
myCRR.typeId = "3"; // For Authors
ReadResponse rr = _service.get(myCRR);
```

Working With 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. A 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.

You cannot use web services to create a custom segment. However, you can use web services to add, delete, and update segment values. You can manipulate values by using the CustomRecord class. The reason this approach works is that every custom segment defined in your system is also a custom record type. Therefore, whenever you create a value for a segment, in essence you are creating an instance of that record type.

A key aspect of working with custom segment values is identification of the appropriate segment. The way you identify the segment varies depending on your endpoint and the operation you are using. For details, see the following topics:

- Identifying Custom Segments and Values
- Example: Dynamic Discovery of Existing Custom Segments
- Example: Adding a Custom Segment Value
- Example: Updating a Custom Segment Value
- Example: Getting a Custom Segment Value by Using Record ID
- Example: Deleting a Custom Segment Value by Using Record ID
- Example: Getting a Custom Segment Value by Using Type Internal ID
- Example: Deleting a Custom Segment Value by Using Type Internal ID

Note: You can also use web services to set values for custom segments that appear as fields on record instances. For details, see the help topic CustomFieldLists for Setting Custom Segment Values.

Identifying Custom Segments and Values

To work with custom segment values, you must know how to identify the appropriate custom segment. Additionally, depending on the task you want to complete, you may have to know how to identify a particular value. For details, see the following sections:

- Identifying Custom Segments
- Identifying Custom Segment Values

Identifying Custom Segments

There are multiple ways to identify a custom segment. If you are using an endpoint that predates 2016.1, there are some limitations as described in the next section.

Endpoint Requirements

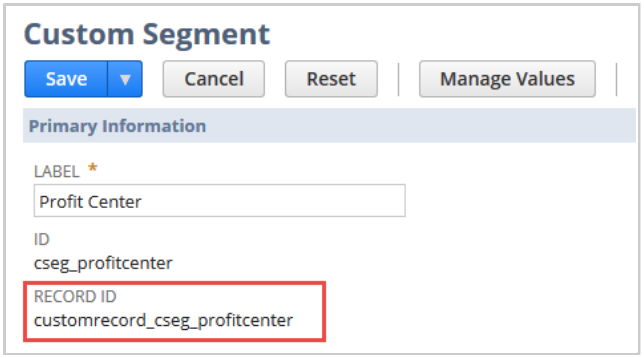
The exact way you identify a custom segment varies depending on the operation and the endpoint you are using. For details, see the following table.

	2013.1 and Earlier	2013.2 - 2015.2	2016.1 and Later
add, update, or upsert	The Custom Record Type's Internal ID	The Custom Segment's Record ID or The Custom Record Type's Internal ID	The Custom Segment's Record ID or The Custom Record Type's Internal ID
get or delete	The Custom Record Type's Internal ID	The Custom Record Type's Internal ID	The Custom Segment's Record ID or The Custom Record Type's Internal ID

The Custom Segment's Record ID

In some cases, you can identify a custom segment by the Record ID value displayed on the segment definition.

You can find the Record ID value by viewing the segment in the UI. Custom segment definitions are available at Customization > Lists, Records, & Fields > Custom Segments. The Record ID field is displayed on the body of the segment.



LABEL *
Profit Center

ID
cseg_profitcenter

RECORD ID
customrecord_cseg_profitcenter

If you are doing an add, update, or upsert, you use this value in conjunction with the CustomizationRef class. Set the CustomizationRef's scriptId as equal to the segment's Record ID. For an example, see Example: Adding a Custom Segment Value.

If you are doing a get or delete, you use this value in conjunction with the CustomRecordRef class. Set the CustomRecordRef's scriptId as equal to the segment's Record ID. For an example, see Example: Getting a Custom Segment Value by Using Record ID.

The Custom Record Type's Internal ID

In some cases, you must identify the custom record type by identifying the custom record type associated with the segment. With this approach, you must use the internal ID associated with that record type. This value cannot be retrieved through the NetSuite UI. However, you can retrieve the internal ID for these custom record types if you interact with them by adding, updating, or upserting a value for the custom segment in conjunction with the 2013.2 endpoint or later. When you do, the SOAP response includes the appropriate internal ID.

The following snippet shows the SOAP response to an update operation. Notice that the response includes a typeId attribute. The value of this attribute (in this example, 39) is the internal ID of the custom record type that corresponds with the custom segment that was identified in the update request.

```
<soapenv:Body>
  <updateResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <writeResponse>
    <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.webservi
ces.netsuite.com"/>
    <baseRef xsi:type="platformCore:CustomRecordRef" typeId="39" internalId="108" xmlns:platfor
mCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
  </writeResponse>
</updateResponse>
```

You can use this value to identify the record type in subsequent interactions with this segment's values. As described in Endpoint Requirements, using this value is the only option for certain endpoints and certain operations.

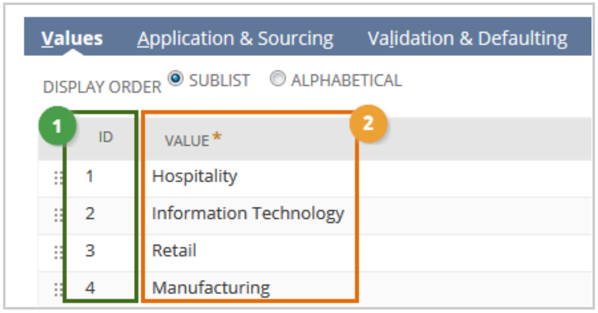
If you are doing an add, update, or upsert, you use this value in conjunction with the RecordRef class. Set the RecordRef's internalId as equal to this value. For an example, see Example: Updating a Custom Segment Value.

If you are doing a get or delete, you use this value in conjunction with the CustomRecordRef class. Set the CustomRecordRef's typeId as equal to this value. For an example, see Example: Getting a Custom Segment Value by Using Type Internal ID.

Identifying Custom Segment Values

If you want to update or delete a specific custom segment value, you must identify it first. You can identify it by using the methods described in the following table and screenshot.

Method	Description	Callout
External ID	A string value that is available only if you created it (by using web services, CSV Import Assistant, or SuiteScript).	
ID	An integer visible when you view the custom segment definition in the UI.	1
Name	A string value visible when you view the custom segment definition in the UI.	2



DISPLAY ORDER [✓ SUBLIST] ALPHABETICAL

ID	VALUE*
1	Hospitality
2	Information Technology
3	Retail
4	Manufacturing

You always identify segment values by using the CustomRecord class.

Example: Dynamic Discovery of Existing Custom Segments

Starting from version 2016.2, you can dynamically discover the record ID of custom segments.

Listing Existing Custom Segments

You can use the getCustomizationId operation with the customRecordType filter to get the list of CustomizationRef values.

SOAP Request

```
<soapenv:Body>
 <getCustomizationId xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <customizationType getCustomizationType="customRecordType"/>
  <includeInactives>false</includeInactives>
 </getCustomizationId>
</soapenv:Body>
```

SOAP Response

```
<soapenv:Body>
   <getCustomizationIdResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <platformCore:getCustomizationIdResult xmlns:platformCore="urn:core_2017_1.platform.webse
rvices.netsuite.com">
         <platformCore:status isSuccess="true"/>
         <platformCore:totalRecords>2</platformCore:totalRecords>
         <platformCore:customizationRefList>
            <platformCore:customizationRef scriptId="customrecord_cseg_example" internalId="24"
 type="customRecordType">
               <platformCore:name>WS Segment Example</platformCore:name>
            </platformCore:customizationRef>
            <platformCore:customizationRef scriptId="customrecord15" internalId="15" type="cust
omRecordType">
               <platformCore:name>Customer Satisfaction Survey</platformCore:name>
            </platformCore:customizationRef>
         </platformCore:customizationRefList>
      </platformCore:getCustomizationIdResult>
   </getCustomizationIdResponse>
</soapenv:Body>
```

Getting Information About the Related Custom Record Type

To get more details about the related custom record type, use the get or getList operation for the CustomRecordType with the references you collected in the previous response.

SOAP Request

```
<soapenv:Body>
 <get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <baseRef internalId="24" scriptId="customrecord_cseg_example" type="customRecordType" xsi:typ
```

```
e="ns7:CustomizationRef" xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">
   <ns7:name xsi:type="xsd:string">WS Segment Example</ns7:name>
  </baseRef>
 </get>
</soapenv:Body>
```

Getting the Values of a Custom Segment

To get the values of a custom segment, use a CustomRecordSearch operation using a reference from the first step as the record type filter.

SOAP Request

```
<soapenv:Body>
 <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <searchRecord xsi:type="ns7:CustomRecordSearchBasic" xmlns:ns7="urn:common_2017_1.platform.we
bservices.netsuite.com">
   <ns7:recType internalId="24" scriptId="customrecord_cseg_example" type="customRecordType" xs
i:type="ns8:CustomizationRef" xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com">
    <ns8:name xsi:type="xsd:string">WS Segment Example</ns8:name>
   </ns7:recType>
  </searchRecord>
 </search>
</soapenv:Body>
```

Example: Adding a Custom Segment Value

This example also shows how to add a value to a custom segment.

The way you identify a custom segment varies depending on the endpoint and operation you are using. This example uses the 2016.1 endpoint. It shows how to identify the segment by using the segment's Record ID value. This method of identifying the segment during an add operation is available only with the 2013.2 endpoint and later.

In this example, the segment has a record ID of customrecord_cseg_profitcenter.

For details on identifying custom segments and their values, see Identifying Custom Segments and Values.

C#

```
private void addSegmentValue()
{

   CustomizationRef myRef = new CustomizationRef();
   myRef.type = RecordType.customRecordType;
   myRef.scriptId = "customrecord_cseg_profitcenter";
   myRef.typeSpecified = true;

   CustomRecord myNewSegmentValue = new CustomRecord();
   myNewSegmentValue.recType = myRef;
   myNewSegmentValue.name = "Apparel";
   myNewSegmentValue.externalId = "Apparel";
   _service.add(myNewSegmentValue);

}
```

SOAP Request

```
<soap:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record externalId="Apparel" xsi:type="q1:CustomRecord" xmlns:q1="urn:customization_2017_
1.setup.webservices.netsuite.com">
         <q1:name>Apparel</q1:name>
         <q1:recType scriptId="customrecord_cseg_profitcenter" type="customRecordType" xsi:type
="q2:CustomizationRef" xmlns:q2="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </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:CustomRecordRef" typeId="39" externalId="Apparel" inte
rnalId="8" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Example: Updating a Custom Segment Value

This example shows how to update an existing custom segment value to include a new name.

The way you update a custom segment value can vary depending on the endpoint and operation you are using. This example uses the 2015.2 endpoint. It demonstrates how to identify the segment by using the internal ID of the appropriate custom record type. This method of identifying the segment is available with all endpoints.

This example also shows how to identify a segment value by its internal ID.

For details on identifying custom segments and their values, see Identifying Custom Segments and Values.

C#

```
  private void updateSegmentValue()
        {

            RecordRef myRef = new RecordRef();
            myRef.type = RecordType.customRecordType;
            myRef.internalId = "39";
            myRef.typeSpecified = true;
            CustomRecord myUpdatedSegmentValue = new CustomRecord();
            myUpdatedSegmentValue.recType = myRef;
            myUpdatedSegmentValue.name = "Furniture";
            myUpdatedSegmentValue.internalId = "108";

            _service.update(myNewSegmentValue);

        }
```

SOAP Request

```
<soap:Body>
   <update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record internalId="108" xsi:type="q1:CustomRecord" xmlns:q1="urn:customization_2017_1.se
tup.webservices.netsuite.com">
         <q1:name>Furniture</q1:name>
         <q1:recType type="customRecordType" internalId="39"/>
      </record>
   </update>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
   <updateResponse 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:CustomRecordRef" typeId="39" internalId="108" xmlns:pl
atformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </updateResponse>
</soapenv:Body>
```

Example: Getting a Custom Segment Value by Using Record ID

This example shows how to get an existing custom segment value.

The way you identify a custom segment value varies depending on the endpoint and operation you are using. This example uses the 2017.1 endpoint. It demonstrates how to identify the segment by using the segment’s Record ID value. This method of identifying a segment during the get operation is available only with the 2016.1 endpoint and later.

For details on identifying custom segments and their values, see Identifying Custom Segments and Values Identifying Custom Segments.

C#

```
private void getSegmentValue()
{

 CustomRecordRef myNewSegmentValue = new CustomRecordRef();
 myNewSegmentValue.scriptId = "customrecord_cseg_profitcenter";
 myNewSegmentValue.externalId = "Apparel";

 _service.get(myNewSegmentValue);
}
```

SOAP Request

```
<soap:Body>
 <get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <baseRef scriptId="customrecord_cseg_profitcenter" externalId="Apparel" xsi:type="q1:CustomRe
cordRef" xmlns:q1="urn:core_2017_1.platform.webservices.netsuite.com">
  </baseRef>
 </get>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
 <getResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
       <readResponse>
          <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.we
bservices.netsuite.com"/>
       <record xsi:type="setupCustom:CustomRecord" externalId="Apparel" internalId="8" xmlns:se
tupCustom="urn:customization_2017_1.setup.webservices.netsuite.com">
      <setupCustom:isInactive>false</setupCustom:isInactive>
         <setupCustom:name>Apparel</setupCustom:name>
         <setupCustom:owner internalId="-4" xmlns:platformCore="urn:core_2017_1.platform.webser
vices.netsuite.com">
                     <platformCore:name>-System-</platformCore:name>
               </setupCustom:owner>
               <setupCustom:recType internalId="39" xmlns:platformCore="urn:core_2017_1.platfor
m.webservices.netsuite.com">
                <platformCore:name>Profit Center</platformCore:name>
              </setupCustom:recType>
          </record>
       </readResponse>
    </getResponse>
</soapenv:Body>
```

Example: Deleting a Custom Segment Value by Using Record ID

This example shows how to delete an existing custom segment value.

The way you identify a custom segment value varies depending on the endpoint and operation you are using. This example uses the 2017.1 endpoint, and the segment is identified by using the segment's Record ID value. This method of identifying a segment during an operation is available only with the 2016.1 endpoint and later.

SOAP Request

```
<soapenv:Body>
```

```
 <delete xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <baseRef externalId="Apparel" scriptId="customrecord_cseg_profitcenter" xsi:type="ns7:CustomR
ecordRef" xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com"/>
 </delete>
</soapenv: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.webservic
es.netsuite.com"/>
   <baseRef internalId="8" externalId="Apparel" scriptId="customrecord_cseg_profitcenter" xsi:t
ype="platformCore:CustomRecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.net
suite.com"/>
  </writeResponse>
 </deleteResponse>
</soapenv:Body>
```

Example: Getting a Custom Segment Value by Using Type Internal ID

This example shows how to get an existing custom segment value.

The way you identify a custom segment value can vary depending on the endpoint and operation you are using. This example uses the 2017.1 endpoint. It demonstrates how to identify the segment by using the internal ID of the custom record type that represents the segment. This method is available with all endpoints. The value is identified by external ID.

For details on identifying custom segments and their values, see Identifying Custom Segments and Values Identifying Custom Segments.

C#

```
private void getSegmentValue()
{

 CustomRecordRef myNewSegmentValue = new CustomRecordRef();
 myNewSegmentValue.typeId = "39";
 myNewSegmentValue.externalId = "Apparel";

 _service.get(myNewSegmentValue);
}
```

SOAP Request

```
<soap:Body>
 <get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  baseRef typeId="39" externalId="Apparel" xsi:type="q1:CustomRecordRef" xmlns:q1="urn:core_201
7_1.platform.webservices.netsuite.com"/>
 </get>
```

```
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
 <getResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
   <readResponse>
     <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.webserv
ices.netsuite.com"/>
     <record xsi:type="setupCustom:CustomRecord" externalId="Apparel" internalId="8" xmlns:setu
pCustom="urn:customization_2017_1.setup.webservices.netsuite.com">
      <setupCustom:isInactive>false</setupCustom:isInactive>
      <setupCustom:name>Apparel</setupCustom:name>
      <setupCustom:owner internalId="-4" xmlns:platformCore="urn:core_2017_1.platform.webservic
es.netsuite.com">
       <platformCore:name>-System-</platformCore:name>
      </setupCustom:owner>
      <setupCustom:recType internalId="39" xmlns:platformCore="urn:core_2017_1.platform.webserv
ices.netsuite.com">
       <platformCore:name>Profit Center</platformCore:name>
     </setupCustom:recType>
    </record>
   </readResponse>
  </getResponse>
</soapenv:Body>
```

Example: Deleting a Custom Segment Value by Using Type Internal ID

This example shows how to delete an existing custom segment value.

The way you identify a custom segment value can vary depending on the endpoint and operation you are using. This example uses the 2017.1 endpoint. It demonstrates how to identify the segment by using the internal ID of the custom record type that represents the segment. This method is available with all endpoints. The value is identified by external ID.

SOAP Request

```
<soapenv:Body>
 <delete xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <baseRef externalId="Apparel" typeId="39" xsi:type="ns7:CustomRecordRef" xmlns:ns7="urn:core_
2017_1.platform.webservices.netsuite.com"/>
 </delete>
</soapenv: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.webservic
```

```
es.netsuite.com"/>
   <baseRef internalId="8" externalId="Apparel" typeId="39" xsi:type="platformCore:CustomRecord
Ref" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
  </writeResponse>
 </deleteResponse>
</soapenv:Body>
```

Custom Record Type

For custom record types, fields on the body and subtabs (Numbering, Permissions, Links, and Managers) can be set on add or update of a new Custom Record Type. Fields on the Fields, Forms, Online Forms, Child Records, and Parent Records subtabs can only be set on update to an existing custom record.

The custom record type record is defined in the setupCustom (customization) XSD.

Supported Operations

The following operations can be used with custom record types.

add | addList | delete | deleteList | getDeleted | get | getCustomizationId | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's custom record type reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Working with Custom Record Sublists

The SuiteTalk Schema Browser includes all sublists associated with the custom record record type. See the following information for usage notes regarding specific Custom Record sublists. Usage notes are not provided for every sublist type.

- CustomRecordTypeChildren
- CustomRecordTypeForms
- CustomRecordTypeOnlineForms
- CustomRecordTypeParents

CustomRecordTypeChildren

This is a **read-only** list that returns all custom records that define the current record as it's parent.

CustomRecordTypeForms

This is a **read-only** list that returns the forms that have been created for use with the current record type. A single default form is automatically created for any new record type.

CustomRecordTypeOnlineForms

This is a **read-only** list that returns all online forms that have been created for the current custom record type.

CustomRecordTypeParents

This is a read-only list that returns the parent records this record is a child of.

To have the entries set here active, you must set the *usePermission* field to TRUE for the record type definition.

Custom Record Custom Field

Custom record custom fields can only be set on a custom record after the custom record has been created.

The custom record custom field record is defined in the setupCustom (customization) XSD.

Supported Operations

The following operations can be used with custom record custom fields.

add | addList | delete | deleteList | get | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

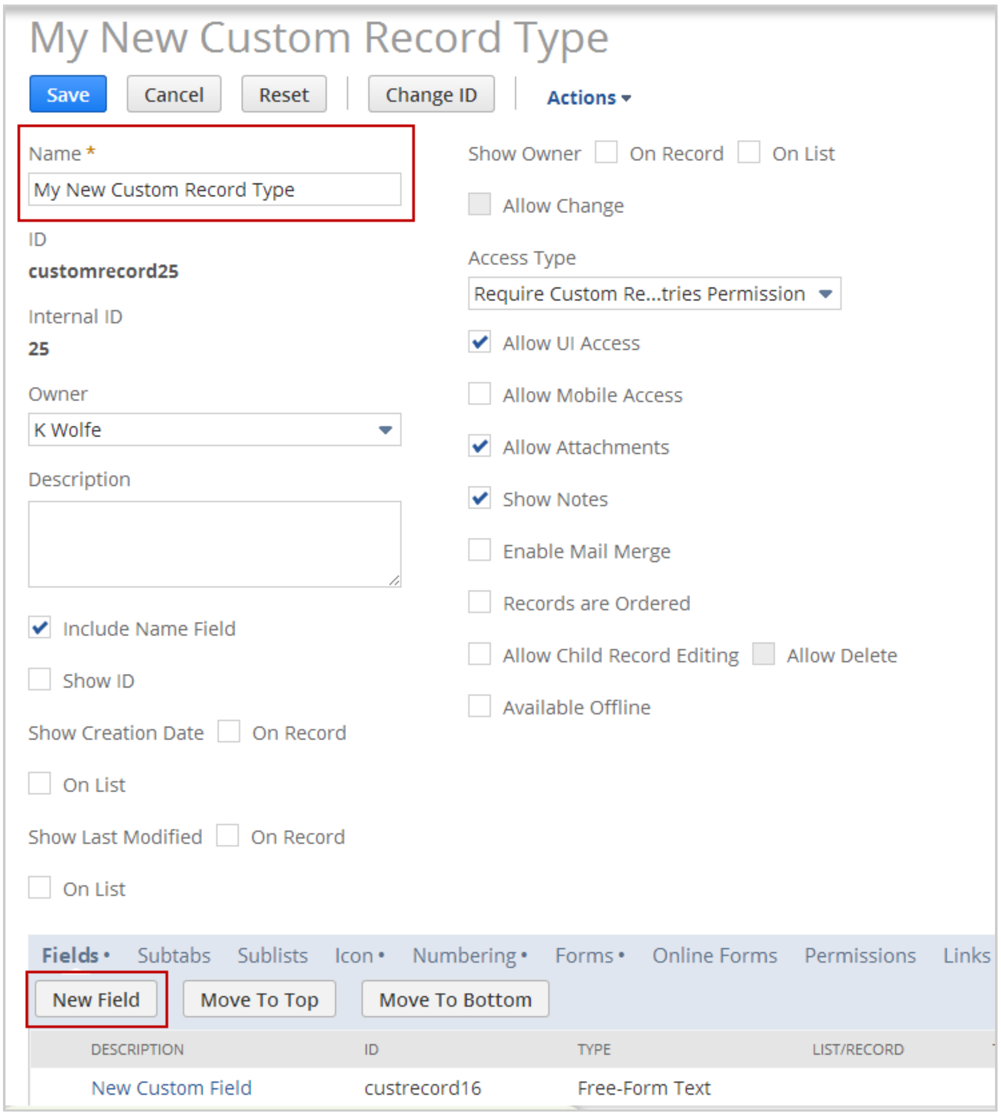
Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's custom record custom field reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

The following figure shows the UI equivalent of creating a custom record custom field. First you must create a new custom record type. After the new type is created, click the New Field button on the Fields tab. This is the only way you can create a custom record custom field, which is directly associated with the custom record type. As a consequence, you cannot directly search for custom record custom fields using either the search() or getCustomization() operations. You must perform a record type search and locate your custom record custom field within that search.



My New Custom Record Type

Name * My New Custom Record Type

ID customrecord25

Internal ID 25

Owner K Wolfe

Description

- [✓] Include Name Field
- [ ] Show ID
- Show Creation Date [ ] On Record [ ] On List
- Show Last Modified [ ] On Record [ ] On List
- Show Owner [ ] On Record [ ] On List
- [ ] Allow Change

Access Type: Require Custom Re...tries Permission

- [✓] Allow UI Access
- [ ] Allow Mobile Access
- [✓] Allow Attachments
- [✓] Show Notes
- [ ] Enable Mail Merge
- [ ] Records are Ordered
- [ ] Allow Child Record Editing
- [ ] Allow Delete
- [ ] Available Offline

Fields • Subtabs Sublists Icon • Numbering • Forms • Online Forms Permissions Links
New Field
Move To Top
Move To Bottom

DESCRIPTION	ID	TYPE	LIST/RECORD
New Custom Field	custrecord16	Free-Form Text	

Custom List

The custom list record is defined in the setupCustom (customization) XSD.

Supported Operations

The following operations can be used with custom list.

add | addList | delete | deleteList | get | getCustomizationId | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's custom list reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

The following Java sample shows how to create a custom list with three values using web services.

```
      CustomListCustomValue Beginner = new CustomListCustomValue();
      Beginner.setValue("Beginner");
      CustomListCustomValue Intermediate = new CustomListCustomValue();
      Intermediate.setValue("Intermediate");
      CustomListCustomValue Advanced = new CustomListCustomValue();
      Advanced.setValue("Advanced");


      CustomListCustomValueList customValueList = new CustomListCustomValueList();
      customValueList.setCustomValue(new CustomListCustomValue[]{Beginner, Intermediate, Advanc
ed});

      CustomList customList = new CustomList();
      customList.setName("IT Proficiency Levels");
      customList.setDescription("List of IT proficiency levels");
      customList.setCustomValueList(customValueList);

      c.addRecord(customList);
```

The following SOAP sample shows how to set a value from a custom list onto a record (in this case a Contact record) using web services.

```
<soapenv:Body>
      <platformMsgs:add
          xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:s0="urn:relationships_2017_1.lists.webservices.netsuite.com"
          xmlns:platformMsgs="urn:messages_2017_1.platform.webservices.netsuite.com"
          xmlns:xsd="http://www.w3.org/2001/XMLSchema"
          xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
         <platformMsgs:record xsi:type="s0:Contact">
            <s0:entityId>Clint Eastwood</s0:entityId>
            <s0:customFieldList>
               <platformCore:customField xsi:type="platformCore:SelectCustomFieldRef"
         scriptId="custentity6">
                  <platformCore:value typeId="7" internalId="2" />
```

```
                </platformCore:customField>
            </s0:customFieldList>
        </platformMsgs:record>
    </platformMsgs:add>
</soapenv:Body>
```

Where:

- scriptId="custentity6" is the name of the custom list as it appears on the Contact record.
- typeId="7" is the custom list internal id (Customization > Lists, Records, & Fields > Lists - Internal ID)
- internalId="2" is the internal id of the second item on the custom list (Customization > Lists, Records, & Fields > Lists > List - ID)

Also note that when working with custom lists in web services, you can update specific values in a custom list using **valueId** as the key field for the CustomListCustomValue sublist and setting replaceAll flag to FALSE.

Custom Transaction

The CustomTransaction complex type lets you interact with instances of existing custom transaction types. For example, if your account included a custom transaction type called Non-Operational Income Entry, you could use web services to create and modify non-operational income entries. This record is defined in the customization XSD.

For details about working with custom transactions in the user interface, see the help topic Custom Transactions.

For more on using web services to interact with custom transactions, see the following:

- Custom Transaction Supported Operations
- Custom Transaction Body and Sublist Fields
- How List Styles Affect Your Integration
- About the CustomTransactionType Enum Value
- Custom Transaction Code Samples

Custom Transaction Supported Operations

The following operations can be used with custom transactions:

add | addList |attach / detach | delete | deleteList | get | getDeleted | getList | getSelectValue | search |searchMore | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Note the following:

- When using the **add** or **update** operation, you use the tranType field to identify the transaction instance's type. TranType is a RecordRef. Set the RecordRef's internal ID to the appropriate value for the desired custom transaction type. For an example, see Example: Adding an Instance of a Basic Custom Transaction Type.
- When using the **attach** operation, you can attach files only.

- When using the **get** operation, you reference existing custom transaction instances by using CustomTransactionRef. This complex type is also used by the attach, delete, and detach operations. For an example, see Example: Getting a Custom Transaction.
- When **searching** for custom transactions, you use the Transaction search objects: TransactionSearchBasic, TransactionSearch, and TransactionSearchAdvanced. When using TransactionSearchBasic to search for custom transaction instances, you can set the type field to _custom to retrieve custom transaction instances. To limit results to instances of a specific custom transaction type, use the RecordType field. Set this field to the value of the transaction type's script ID.
- In web services, the transaction status filter does not support statuses of custom transactions. This behavior is different from the UI, where the transaction status filter does support custom transactions' statuses. To use custom transaction status to filter a web services search, you need to do the following:
    1. Create a new custom transaction body field with a Type of List/Record, and a List/Record value of Transaction Status.
    2. Create one or more user event scripts to store the proper value in the new transaction body field, and to apply this custom field to the transaction types for which you want to use this field to filter searches.
    3. Run a mass update to set the value for existing records.

  After these steps, SearchMultiSelectCustomField can be used as a filter in web services, as shown in the following code sample.

  ```
  <ns8:customFieldList xsi:type="ns11:SearchCustomFieldList" smlns:ns11="urn:core_2017_1.platform
  .webservices.netsuite.com"> <ns11:customField scriptId="custbody_st" operator="anyof" xsi:type=
  "ns11:SearchMultiSelectCustomField">
  <ns11:searchValue internalId="63" xsi:type="ns11:ListOrRecordRef" />
  <ns11:searchValue internalId="10007" xsi:type="ns11:ListOrRecordRef"/>
  <ns11:searchValue internalId="86" xsi:type="ns11:ListOrRecordRef"/> </ns8:customFieldList>
  ```

  You can use the getSelectValue operation to get the correct numbers for the filter, as shown in the following code sample.

  ```
  <getSelectValue xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <fieldDescription>
  <ns6:recordType xmlns:ns6=urn:core_2017_1.platform.webservices.netsuite.com">invoice</ns6:recor
  dType>
  <ns7:field xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">custboty_st</ns7:field
  >
  </fieldDescription>
  <pageIndex>1</pageIndex>
  </getSelectValue>
  ```

Custom Transaction Body and Sublist Fields

When you work with custom transactions, the available fields vary depending on the transaction type, the features enabled in your account, and other factors. As a best practice, view the entry form for the transaction instance in the UI to see which fields are available. Alternatively, you can open the transaction type for editing.

The following factors affect field availability:

- Depending on the custom transaction type's configuration, the class, department, and location fields may or may not be available. Similarly, these fields can be configured to appear on either the transaction body or Lines sublist. For details on these configuration choices, see the help topic Custom Transaction Type Classification Fields.

- The transaction type may or may not be configured to use the Status field. You can interact with the Status field only if statuses exist, and only if the Show Status Field option has been selected for the custom transaction type. When the field is available, you can import a status by referencing the status's ID. You can view the ID value on the transaction type's Statuses sublist. Note that the ID value is always a letter value between A and Z. Map the ID to the recordRef's internal ID field. For an example, see Example: Adding an Instance of a Journal Custom Transaction Type.
- Different custom transaction types can be configured to use different custom fields. For details on custom field usage, see the help topic Custom Fields in Custom Transaction Types.
- A custom transaction type can have any of three list styles. The list style has an impact on whether the Lines sublist is available and how you use the sublist. For more details, see How List Styles Affect Your Integration.
- The Lines sublist is keyed. You can use Line number as the key. Note that this behavior differs from CSV Import, where you can also use Account as a key, if every line has a unique account value. In web services, you can use only the Line number as the key.
- The accountingBookDetailList sublist is usable only when the Multi-Book Accounting feature is enabled, and only when more than one active accounting book exists. This sublist is keyed. The key is the accountingBook value.
- The subsidiary field is available only for OneWorld accounts.

Note: For full details about all fields exposed as part of the CustomTransaction complex type, refer to the custom transaction reference page in the SuiteTalk Schema Browser. For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

How List Styles Affect Your Integration

A custom transaction type can have any of three list styles. The list style dictates which fields are available when you add or update a custom transaction. For more details, see the following sections:

- Basic
- Journal
- Header Only

Basic

When working with an instance of the basic transaction type, be aware of the following:

- Each transaction must have at least one line in the LineList.
- Each line must include a value for the **account** field and a value for the **amount** field. The account field identifies the account that is to be debited or credited by this transaction. (Whether the account is debited or credited depends on how the transaction type is configured. For details, see the help topic Setting the Account Field for a Basic Transaction Type.) The amount field identifies the exact value of the credit or debit.
- With this style, the lines do not use the credit or debit fields. Values entered for these fields are ignored.

For more information about the basic list style, see the help topic Basic.

Journal

When working with an instance of the journal style transaction type, be aware of the following:

- The memo body field is not available. However, you can set the memo field on any line in the LineList.
- Each transaction instance must have at least two lines in the LineList. More lines are permitted, but at least one line must set a value for the **credit** field and one must set a value for the **debit** field. The values of the lines must balance.
- With this style, the lines do not use the amount field. Values entered for these fields are ignored.

For more information about the journal list style, see the help topic Journal.

Header Only

With the header only style, it is not possible to interact with the LineList. The only lines associated with these transactions are the lines generated by a custom GL Lines plug-in implementation. For more details, see the help topic Header Only.

About the CustomTransactionType Enum Value

SuiteTalk includes a complex type to represent custom transactions, but not custom transaction types. What this means is that you can use web services to create, edit, or get custom transaction entries – instances of existing custom transaction types. However, you cannot create, edit, or get the types themselves. Custom transaction types can be created and edited only through the UI and SuiteBundler.

However, you must refer to custom transaction types when completing certain tasks in web services. To support these capabilities, the RecordType and GetCustomizationType enumerations each include customTransactionType as a value. You can use this value to accomplish tasks such as the following:

- Get a List of Valid Custom Transaction Types
- Create a Custom Transaction Instance
- Retrieve a List of Valid Values for a Field on a Custom Transaction Instance

Get a List of Valid Custom Transaction Types

There may be times when you need to retrieve a list of valid custom transaction types that exist in your account. To retrieve this list, use the GetCustomizationType enumeration and the getCustomizationId operation, as shown in the following C# snippet:

```
private void getCustomizationID()
{
   CustomizationType myType = new CustomizationType();
   myType.getCustomizationType = GetCustomizationType.customTransactionType;
   myType.getCustomizationTypeSpecified = true;

   GetCustomizationIdResult getCustId = _service.getCustomizationId(myType, false);
}
```

In this example, the GetCustomizationType enumeration tells NetSuite which type of customization you need information about. In its response, the system returns a list of CustomizationRef objects, one for each custom transaction type that exists:

```
<soapenv:Body>
   <getCustomizationIdResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <platformCore:getCustomizationIdResult xmlns:platformCore="urn:core_2017_1.platform.webse
rvices.netsuite.com">
         <platformCore:status isSuccess="true"/>
```

```
        <platformCore:totalRecords>2</platformCore:totalRecords>
        <platformCore:customizationRefList>
           <platformCore:customizationRef type="customTransactionType" internalId="100" script
Id="customtransaction_baddebt">
              <platformCore:name>Bad Debt</platformCore:name>
           </platformCore:customizationRef>
           <platformCore:customizationRef type="customTransactionType" internalId="101" script
Id="customtransaction_fixedassetdepreciation">
              <platformCore:name>Fixed Asset Depreciation</platformCore:name>
           </platformCore:customizationRef>
        </platformCore:customizationRefList>
     </platformCore:getCustomizationIdResult>
  </getCustomizationIdResponse>
</soapenv:Body>
```

Create a Custom Transaction Instance

When you create a custom transaction instance, you can use the RecordType enumeration, as shown in the following snippet:

```
   RecordRef myCustTrxnType = new RecordRef();
   myCustTrxnType.type = RecordType.customTransactionType;
   myCustTrxnType.typeSpecified = true;
   myCustTrxnType.internalId = "100";
   myCustomTrxn.tranType = myCustTrxnType;
```

In this example, the use of RecordType.customTransactionType tells the system that you are creating an instances of a custom transaction type. The internal ID value tells the system which specific custom transaction type you want to create an instance of.

For a full example of how to add a custom transaction instance, see Example: Adding an Instance of a Basic Custom Transaction Type.

Retrieve a List of Valid Values for a Field on a Custom Transaction Instance

When you create an instance of a custom transaction type, the entry form for the type may include a dropdown list. At times, you may need to retrieve a list of possible values for this type of field. You can retrieve this list by using the RecordType enumeration and the getSelectValue operation, as shown in the following snippet:

```
private void getSelectValue()
{

CustomizationRef newRef = new CustomizationRef();
newRef.typeSpecified = true;
newRef.type = RecordType.customTransactionType;
newRef.internalId = "100";

GetSelectValueFieldDescription Request_fields = new GetSelectValueFieldDescription();
Request_fields.customTransactionType = newRef;
Request_fields.field = "subsidiary";
GetSelectValueResult result = _service.getSelectValue(Request_fields, 0);
BaseRef[] baseRef = result.baseRefList;
}
```

In this example, note the following:

- The value RecordType.customTransactionType indicates that the request is related to a custom transaction type.
- The internal ID field identifies the exact custom transaction type that is of concern.
- The value of Request_fields.field tells NetSuite that you are looking for a list of valid values for the Subsidiary field.

Custom Transaction Code Samples

See the following for examples of how to structure your code:

- Example: Getting a Custom Transaction
- Example: Adding an Instance of a Basic Custom Transaction Type
- Example: Adding an Instance of a Journal Custom Transaction Type

Example: Getting a Custom Transaction

In some cases, you may want to retrieve a custom transaction instance by using the get operation. When doing a get of most record types, you use the RecordRef class. However, with a custom transaction, you use CustomTransactionRef class.

When creating a CustomTransactionRef object, you identify the custom transaction type by using either the scriptId field or the typeId field. Both fields reference fields on the custom transaction type definition: The scriptId field identifies the field labeled ID in the UI. The typeId field takes the internal ID of the custom transaction type.

By contrast, the CustomTransactionRef's internal ID field is used to identify the internal ID of the desired transaction instance.

The following sample code shows how to get an instance of the custom transaction type with the script ID customtransaction_fixedasset. The operation retrieves the instance with the internal ID of 9675.

C#

```
private void getCustomTrxn()
{

   CustomTransactionRef customTrxn = new CustomTransactionRef();
   customTrxn.scriptId = "customtransaction_fixedasset";
   customTrxn.internalId = "9675";
   ReadResponse response = _service.get(customTrxn);

}
```

SOAP Request

```
<soap:Body>
   <get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <baseRef scriptId="customtransaction_fixedasset" internalId="9675" xsi:type="q1:CustomTra
nsactionRef" xmlns:q1="urn:core_2017_1.platform.webservices.netsuite.com"/>
```

```
   </get>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
   <getResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <readResponse>
         <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.web
services.netsuite.com"/>
         <record xsi:type="setupCustom:CustomTransaction" internalId="9675" xmlns:setupCustom="
urn:customization_2017_1.setup.webservices.netsuite.com">
         <setupCustom:createdDate>2015-03-20T06:09:47.000-07:00</setupCustom:createdDate>
         <setupCustom:lastModifiedDate>2015-03-20T06:09:47.000-07:00</setupCustom:lastModifiedD
ate>
         <setupCustom:tranType internalId="110" xmlns:platformCore="urn:core_2017_1.platform.we
bservices.netsuite.com">
            <platformCore:name>Fixed Asset Accrual</platformCore:name>
         </setupCustom:tranType>
         <setupCustom:tranId>1</setupCustom:tranId>
         <setupCustom:currency internalId="1" xmlns:platformCore="urn:core_2017_1.platform.webs
ervices.netsuite.com">
            <platformCore:name>US Dollar</platformCore:name>
         </setupCustom:currency>
         <setupCustom:exchangeRate>1.0</setupCustom:exchangeRate>
         <setupCustom:tranDate>2015-03-20T00:00:00.000-07:00</setupCustom:tranDate>
         <setupCustom:postingPeriod internalId="351" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com">
            <platformCore:name>Mar 2015</platformCore:name>
         </setupCustom:postingPeriod>
         <setupCustom:tranStatus internalId="A" xmlns:platformCore="urn:core_2017_1.platform.we
bservices.netsuite.com">
            <platformCore:name>Approved</platformCore:name>
         </setupCustom:tranStatus>
         <setupCustom:subsidiary internalId="1" xmlns:platformCore="urn:core_2017_1.platform.we
bservices.netsuite.com">
            <platformCore:name>Parent Company</platformCore:name>
         </setupCustom:subsidiary>
         <setupCustom:lineList>
            <setupCustom:customTransactionLine>
               <setupCustom:account internalId="6" xmlns:platformCore="urn:core_2017_1.platform
.webservices.netsuite.com">
                  <platformCore:name>10100 Investments</platformCore:name>
               </setupCustom:account>
               <setupCustom:debit>1600.0</setupCustom:debit>
            </setupCustom:customTransactionLine>
            <setupCustom:customTransactionLine>
               <setupCustom:account internalId="12" xmlns:platformCore="urn:core_2017_1.platfor
m.webservices.netsuite.com">
                  <platformCore:name>10300 Undeposited Funds</platformCore:name>
               </setupCustom:account>
               <setupCustom:credit>1600.0</setupCustom:credit>
            </setupCustom:customTransactionLine>
         </setupCustom:lineList>
```

```
          <setupCustom:accountingBookDetailList>
             <setupCustom:customTransactionAccountingBookDetail>
                <setupCustom:accountingBook internalId="5" xmlns:platformCore="urn:core_2017_1.p
latform.webservices.netsuite.com">
                   <platformCore:name>Jan</platformCore:name>
                </setupCustom:accountingBook>
                <setupCustom:currency internalId="1" xmlns:platformCore="urn:core_2017_1.platfor
m.webservices.netsuite.com">
                   <platformCore:name>US Dollar</platformCore:name>
                </setupCustom:currency>
                <setupCustom:exchangeRate>1.0</setupCustom:exchangeRate>
           </setupCustom:customTransactionAccountingBookDetail>
          </setupCustom:accountingBookDetailList>
     </record>
    </readResponse>
</getResponse>
     </soapenv:Body>
```

Example: Adding an Instance of a Basic Custom Transaction Type

The following example shows how to add a custom transaction with two lines. In this example, the custom transaction type has a list style of Basic. For that reason, each sublist line uses the **amount** field for the monetary value.

C#

```
private void addCustomTransaction()
{

   CustomTransaction myCustomTrxn = new CustomTransaction();
   myCustomTrxn.externalId = "1052A";


   // Create a RecordRef to identify the custom transaction type

   RecordRef myCustTrxnType = new RecordRef();
   myCustTrxnType.type = RecordType.customTransactionType;
   myCustTrxnType.typeSpecified = true;
   myCustTrxnType.internalId = "109";
   myCustomTrxn.tranType = myCustTrxnType;


   // If this is a OneWorld account, create a RecordRef to
   //identify the subsidiary

   RecordRef mySubsidiary = new RecordRef();
   mySubsidiary.internalId = "1";
   myCustomTrxn.subsidiary = mySubsidiary;


   // Create a sublist and add two lines.

   CustomTransactionLineList myLineList = new CustomTransactionLineList();
   myLineList.customTransactionLine = new CustomTransactionLine[2];
   myCustomTrxn.lineList = myLineList;


   myLineList.customTransactionLine[0] = new CustomTransactionLine();

   RecordRef myAccount = new RecordRef();
   myAccount.type = RecordType.account;
   myAccount.typeSpecified = true;
   myAccount.internalId = "9";

   myLineList.customTransactionLine[0].account = myAccount;
   myLineList.customTransactionLine[0].amount = 1000;
   myLineList.customTransactionLine[0].amountSpecified = true;

   myLineList.customTransactionLine[1] = new CustomTransactionLine();

   RecordRef myAccount2 = new RecordRef();
   myAccount2.type = RecordType.account;
   myAccount2.typeSpecified = true;
   myAccount2.internalId = "10";

   myLineList.customTransactionLine[1].account = myAccount2;
   myLineList.customTransactionLine[1].amount = 2000;
   myLineList.customTransactionLine[1].amountSpecified = true;


   // Add the custom transaction instance.

   _service.add(myCustomTrxn);

}
```

SOAP Request

```
<soap:Body>
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record externalId="1053A" xsi:type="q1:CustomTransaction" xmlns:q1="urn:customization_20
17_1.setup.webservices.netsuite.com">
         <q1:tranType type="customTransactionType" internalId="109"/>
         <q1:subsidiary internalId="1"/>
         <q1:lineList>
            <q1:customTransactionLine>
               <q1:account type="account" internalId="9"/>
               <q1:amount>1000</q1:amount>
            </q1:customTransactionLine>
            <q1:customTransactionLine>
               <q1:account type="account" internalId="10"/>
               <q1:amount>2000</q1:amount>
            </q1:customTransactionLine>
         </q1:lineList>
      </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="customTransaction" externalId="1053A"
 internalId="9579" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Example: Adding an Instance of a Journal Custom Transaction Type

The following example shows how to add a custom transaction with two lines. In this example, the custom transaction type has a list style of Journal. For that reason, each sublist line uses either the **credit** or **debit** field for the monetary value. Each transaction must have at least one credit line and one debit line, and the lines must balance.

Note that when you create an instance of a journal custom transaction type, the memo body field is not available.

C#

```
private void addCustomTransaction()
{

   CustomTransaction myCustomTrxn = new CustomTransaction();
   myCustomTrxn.externalId = "1055A";


   // Create a RecordRef to identify the custom transaction type

   RecordRef myCustTrxnType = new RecordRef();
   myCustTrxnType.type = RecordType.customTransactionType;
   myCustTrxnType.typeSpecified = true;
   myCustTrxnType.internalId = "110";
   myCustomTrxn.tranType = myCustTrxnType;


   // If this is a OneWorld account, create a RecordRef to
   //identify the subsidiary

   RecordRef mySubsidiary = new RecordRef();
   mySubsidiary.internalId = "1";
   myCustomTrxn.subsidiary = mySubsidiary;


   // If the transaction type includes other required body fields,
   // set values for them.

   RecordRef myClass = new RecordRef();
```

```
    myClass.internalId = "6";
    myCustomTrxn.@class = myClass;


    // If the transaction type includes a status field, set a value
    // for it, as appropriate. Use a RecordRef, and set the internalId
    // field to the ID value of the status, as shown in the UI on the
    // Statuses sublist.

    RecordRef myStatus = new RecordRef();
    myStatus.internalId = "A";
    myCustomTrxn.tranStatus = myStatus;


    // Create a sublist and two lines.

    CustomTransactionLineList myLineList = new CustomTransactionLineList();
    myLineList.customTransactionLine = new CustomTransactionLine[2];
    myCustomTrxn.lineList = myLineList;


    myLineList.customTransactionLine[0] = new CustomTransactionLine();

    RecordRef myDebitAccount = new RecordRef();
    myDebitAccount.type = RecordType.account;
    myDebitAccount.typeSpecified = true;
    myDebitAccount.internalId = "6";

    myLineList.customTransactionLine[0].account = myDebitAccount;
    myLineList.customTransactionLine[0].debit = 1600;
    myLineList.customTransactionLine[0].debitSpecified = true;


    myLineList.customTransactionLine[1] = new CustomTransactionLine();

    RecordRef myCreditAccount = new RecordRef();
    myCreditAccount.type = RecordType.account;
    myCreditAccount.typeSpecified = true;
    myCreditAccount.internalId = "12";

    myLineList.customTransactionLine[1].account = myCreditAccount;
    myLineList.customTransactionLine[1].credit = 1600;
    myLineList.customTransactionLine[1].creditSpecified = true;


    // Add the custom transaction instance.

    _service.add(myCustomTrxn);

}
```

SOAP Request

```
<soap:Body>
```

```
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record externalId="1055A" xsi:type="q1:CustomTransaction" xmlns:q1="urn:customization_20
17_1.setup.webservices.netsuite.com">
         <q1:tranType type="customTransactionType" internalId="110"/>
         <q1:tranStatus internalId="A"/>
         <q1:subsidiary internalId="1"/>
         <q1:class internalId="6"/>
         <q1:lineList>
            <q1:customTransactionLine>
               <q1:account type="account" internalId="6"/>
               <q1:debit>1600</q1:debit>
            </q1:customTransactionLine>
            <q1:customTransactionLine>
               <q1:account type="account" internalId="12"/>
               <q1:credit>1600</q1:credit>
            </q1:customTransactionLine>
         </q1:lineList>
      </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="customTransaction" externalId="1055A"
 internalId="9676" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

CRM Custom Field

The CRM custom field record is defined in the setupCustom (customization) XSD.

Supported Operations

The following operations can be used with CRM custom field.

add | addList | delete | deleteList | get | getCustomizationId | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's CRM custom field reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Working with CRM Custom Fields Sublists

The SuiteTalk Schema Browser includes all sublists associated with the CRM custom fields record. See the following information for usage notes regarding specific CRM custom fields sublists. Usage notes are not provided for every sublist type.

CrmCustomFieldFilter

This list can only be populated when the type of custom field is set to List/Record.

Entity Custom Field

The entity custom field record is defined in the setupCustom (customization) XSD.

Supported Operations

The following operations can be used with entity custom field.

add | addList | delete | deleteList | get | getCustomizationId | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's entity custom field reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Item Custom Field

The item custom field record is defined in the setupCustom (customization) XSD.

Supported Operations

The following operations can be used with item custom field.

add | addList | delete | deleteList | get | getCustomizationId | getList | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's item custom field reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Item Number Custom Field

The item number custom field record is defined in the setupCustom (customization) XSD.

Supported Operations

The following operations can be used with item number custom field.

add | addList | delete | deleteList | get | getCustomizationId | getList | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's item number custom field reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Item Option Custom Field

The item option custom field record is defined in the setupCustom (customization) XSD.

Supported Operations

The following operations can be used with item option custom field.

add | addList | delete | deleteList | get | getCustomizationId | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's item option custom field reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Other Custom Field

The other custom field record is defined in the setupCustom (customization) XSD.

Supported Operations

The following operations can be used with other custom field.

add | addList | delete | deleteList | get | getCustomizationId | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's other custom field reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Transaction Body Custom Field

The transaction body custom field record is defined in the setupCustom (customization) XSD.

Supported Operations

The following operations can be used with transaction body custom field.

add | addList | delete | deleteList | get | getCustomizationId | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's transaction body custom field reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Transaction Column Custom Field

The transaction column custom field record is defined in the setupCustom (customization) XSD.

Supported Operations

The following operations can be used with transaction column custom field.

add | addList | delete | deleteList | get | getCustomizationId | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's transaction column custom field reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Code Sample

The following sample shows how to set transaction column custom fields.

SOAP

```
<soapenv:Body>
<add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<record xsi:type="ns9:SalesOrder" xmlns:ns9="urn:sales_2017_1.transactions.webservices.netsuite
.com">
  <ns9:entity internalId="2" xsi:type="ns10:RecordRef" xmlns:ns10="urn:core_2017_1.platform.web
services.netsuite.com" />
<ns9:itemList replaceAll="false" xsi:type="ns9:SalesOrderItemList">
<ns9:item xsi:type="ns9:SalesOrderItem">
  <ns9:item internalId="34" xsi:type="ns11:RecordRef" xmlns:ns11="urn:core_2017_1.platform.webs
ervices.netsuite.com" />
  <ns9:quantity xsi:type="xsd:double">3.0</ns9:quantity>
<ns9:customFieldList xsi:type="ns12:CustomFieldList" xmlns:ns12="urn:core_2017_1.platform.webse
rvices.netsuite.com">
<ns12:customField scriptId="custcol_my_bool" xsi:type="ns12:BooleanCustomFieldRef">
  <ns12:value xsi:type="xsd:boolean">true</ns12:value>
  </ns12:customField>
<ns12:customField scriptId="custcol_my_string" xsi:type="ns12:StringCustomFieldRef">
  <ns12:value xsi:type="xsd:string">hello world</ns12:value>
  </ns12:customField>
<ns12:customField scriptId="custcol_my_integer" xsi:type="ns12:LongCustomFieldRef">
  <ns12:value xsi:type="xsd:long">100</ns12:value>
  </ns12:customField>
  </ns9:customFieldList>
  </ns9:item>
  </ns9:itemList>
  </record>
  </add>
  </soapenv:Body>
```

Java

```
SalesOrder so = new SalesOrder();

RecordRef entityRef = new RecordRef();
entityRef.setType(RecordType.customer);
entityRef.setInternalId("2");
so.setEntity(entityRef);

SalesOrderItem soi = new SalesOrderItem();

RecordRef itemRef = new RecordRef();
itemRef.setType(RecordType.inventoryItem);
itemRef.setInternalId("34");
soi.setItem(itemRef);
soi.setQuantity(new Double(3));

BooleanCustomFieldRef cf1 = new BooleanCustomFieldRef(true, "custcol_my_bool");
StringCustomFieldRef cf2 = new StringCustomFieldRef("hello world", "custcol_my_string");
LongCustomFieldRef cf3 = new LongCustomFieldRef(100, "custcol_my_integer");

soi.setCustomFieldList(new CustomFieldList(new CustomFieldRef[]{cf1, cf2, cf3}));
so.setItemList(new SalesOrderItemList(new SalesOrderItem[]{soi}, true));
```

Marketing Records

The following marketing records are supported in SuiteTalk:

- Campaign
- Campaign Category
- Campaign Audience
- Campaign Family
- Campaign Search Engine
- Campaign Channel
- Campaign Offer
- Campaign Response
- Campaign Vertical
- Campaign Subscription
- Coupon Code
- Promotion Code

Campaign

The campaign record is defined in the listMkt (marketing) XSD.

Supported Operations

The following operations can be used with campaign records:

add | addList | attach / detach | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's campaign reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Campaign Category

The campaign category record is defined in the listMkt (marketing) XSD.

Supported Operations

The following operations can be used with campaign category records:

add | addList | delete | deleteList | get | getAll | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's campaign category reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Campaign Audience

The campaign audience record is defined in the listMkt (marketing) XSD.

Supported Operations

The following operations can be used with campaign audience records:

add | addList | delete | deleteList | get | getAll | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's campaign audience reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Campaign Family

The campaign family record is defined in the listMkt (marketing) XSD.

Supported Operations

The following operations can be used with campaign family records:

add | addList | delete | deleteList | get | getAll | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's campaign family reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Campaign Search Engine

The campaign search engine record is defined in the listMkt (marketing) XSD.

Supported Operations

The following operations can be used with campaign search engine records:

add | addList | delete | deleteList | get | getAll | getList | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's campaign search engine reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Campaign Channel

The campaign channel record is defined in the listMkt (marketing) XSD.

Supported Operations

The following operations can be used with campaign channel records:

add | addList | delete | deleteList | get | getAll | getList | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's campaign channel reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Campaign Offer

The campaign offer record is defined in the listMkt (marketing) XSD.

Supported Operations

The following operations can be used with campaign offer records:

add | addList | delete | deleteList | get | getAll | getList | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's campaign offer reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Campaign Response

The campaign response record helps you refine how you deliver marketing campaigns.

To view this record in the user interface, navigate to an entity record and click the Marketing subtab. On the Campaigns subtab, click the Add Response button.

For help working with this record manually, see the help topic Tracking Campaign Responses.

This record is defined in listMkt (marketing) XSD, where it is called CampaignResponse.

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's campaign response reference page. For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

For more on using web services to interact with the campaign response record, see the following:

- Campaign Response Supported Operations
- Working with the campaignResponseDate Field
- Campaign Response Usage Notes
- Example — Adding a Campaign Response Record
- Example — Getting a Campaign Response Record

Campaign Response Supported Operations

The following operations can be used with campaign response records:

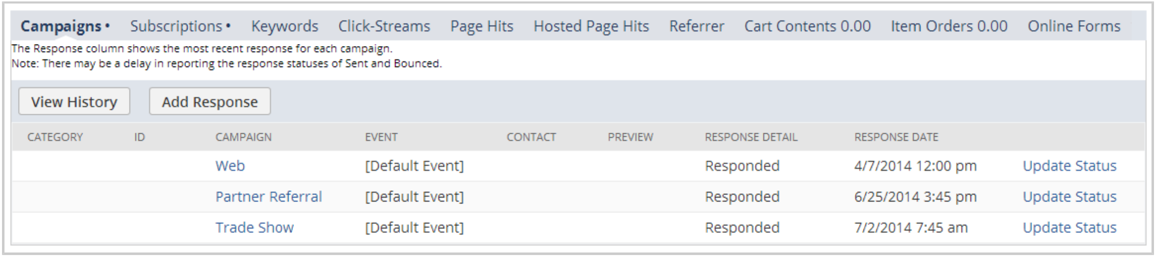
add | addList | get | getList | getSelectValue | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Behavior of the Add and Update Operations

The information on a campaign response record cannot be changed, either in the user interface or when using web services. The only change possible is to add lines to the record's sublist. Further, if you attempt to add a record using a combination of fields that is not unique, the system does not add a new record. It adds a sublist line to an existing record.

To better understand this, consider how these records are created and represented in the user interface. Campaign responses are listed on the entity record with which they are associated. Specifically, they are listed on the Campaigns subtab of the Marketing subtab.

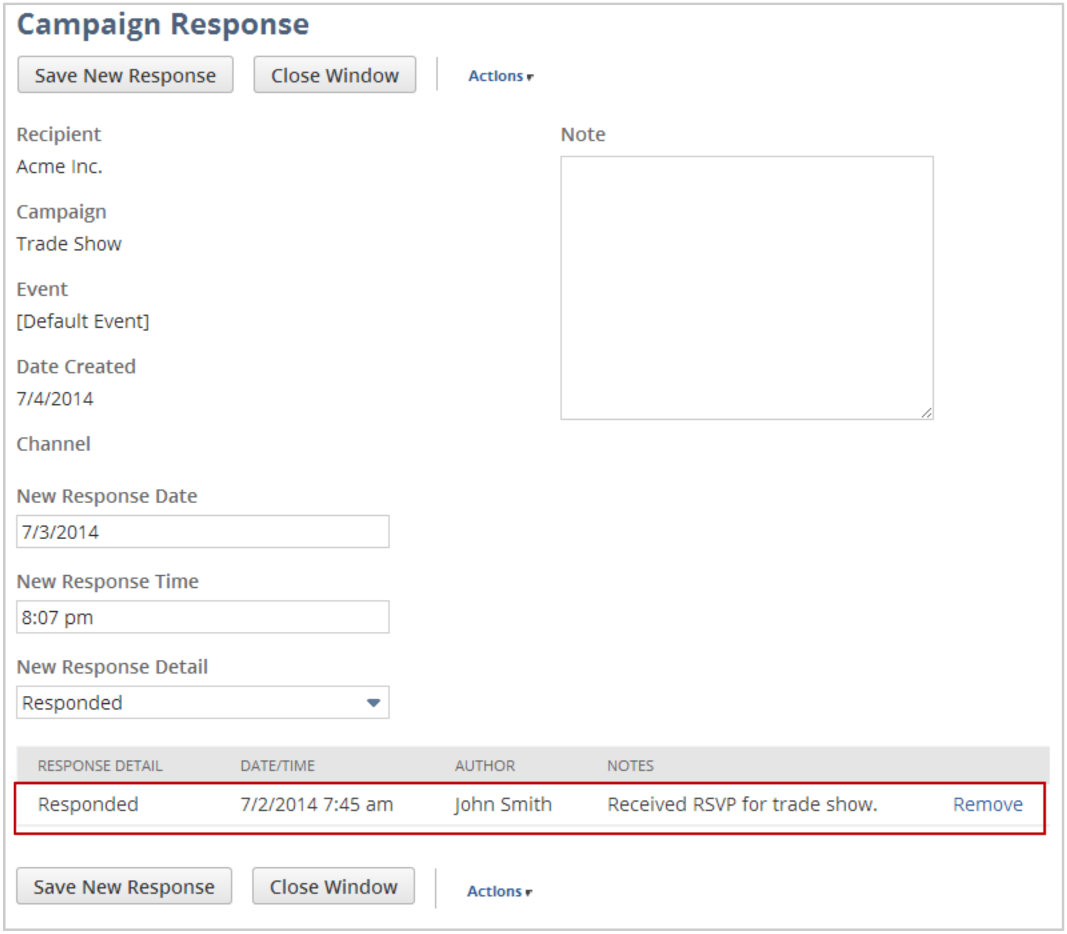


The Response column shows the most recent response for each campaign.
Note: There may be a delay in reporting the response statuses of Sent and Bounced.

CATEGORY	ID	CAMPAIGN	EVENT	CONTACT	PREVIEW	RESPONSE DETAIL	RESPONSE DATE	
		Web	[Default Event]			Responded	4/7/2014 12:00 pm	Update Status
		Partner Referral	[Default Event]			Responded	6/25/2014 3:45 pm	Update Status
		Trade Show	[Default Event]			Responded	7/2/2014 7:45 am	Update Status

When you create a campaign response for a campaign/event combination that is not already represented the entity's subtab, the system adds a line of data, along with an Update Status link. For any entity, a particular campaign/event combination can be represented only one time on this subtab. For example, in the preceding screenshot, three campaign responses are listed. Note that each has a unique campaign/event combination.

If you click the Update Status link, the system displays the Campaign Response window. This window includes a sublist labeled Response Detail. This sublist includes a line showing the original details you specified when you created the record (shown in the red box in the following screenshot). The window also includes a form for entering more details.



Recipient
Acme Inc.

Campaign
Trade Show

Event
[Default Event]

Date Created
7/4/2014

Channel

New Response Date
7/3/2014

New Response Time
8:07 pm

New Response Detail
Responded

Note

RESPONSE DETAIL	DATE/TIME	AUTHOR	NOTES	
Responded	7/2/2014 7:45 am	John Smith	Received RSVP for trade show.	Remove

If you enter new data on the form and click Save New Response, the system adds a new line to the Response Detail sublist. Similarly, if you cancel out of this screen and click the Add Response button on the Campaigns tab, but you use the form to specify the same campaign and event as an existing campaign response record, the system adds a new line to the Response Detail sublist of the existing record. It does not create a new line on the Campaigns subtab. For example, in the illustration below, two new lines have been added to an existing record.

SuiteTalk mimics this behavior: Any update you make to an existing record results in a new line on the existing campaign response record. Existing data is not changed. Similarly, if you use the add operation but name the same combination of entity, campaign, and event as has already been specified on an existing campaign response, the system adds a line to that same record. It does not create a new record.

Note also that one field — the campaignResponseDate field — is treated differently depending on whether you are doing an add, update, or get operation. For details, see Working with the campaignResponseDate Field.

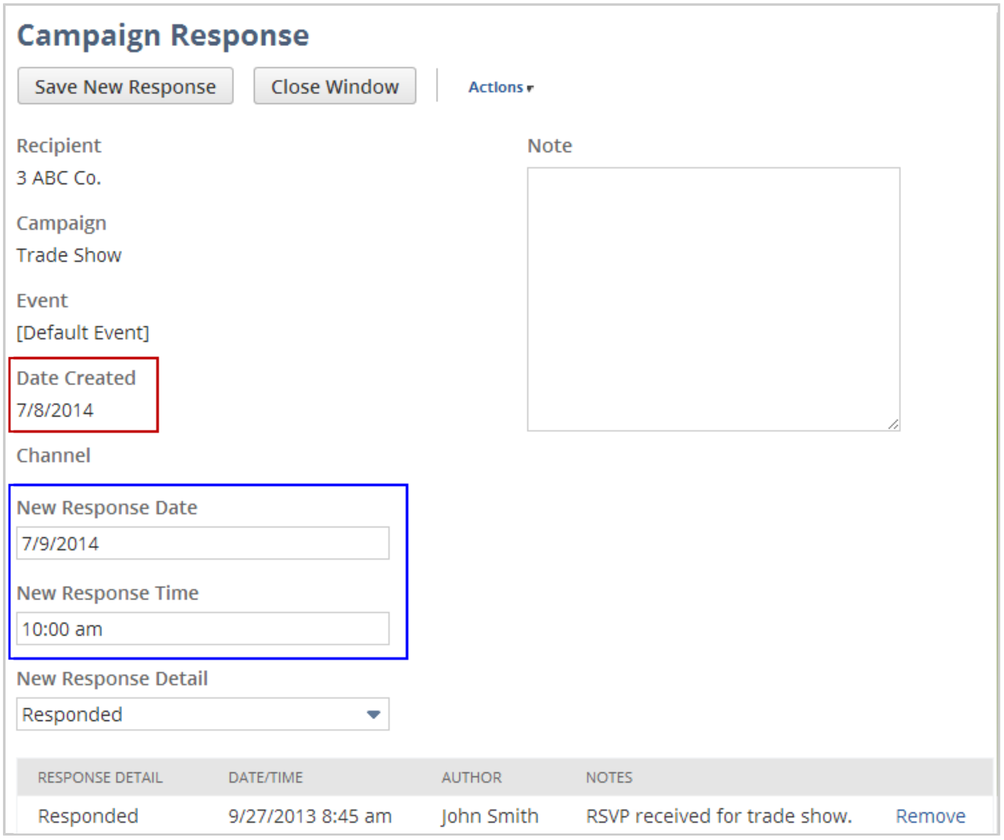
Remove Link Is Not Supported in SuiteTalk

The UI for this record includes a Remove link, which is available for those records that were created manually in the UI or through web services (in contrast to responses created automatically by the system, which do not have this link). However, you can never use SuiteTalk to remove campaign responses, regardless of how they were added.

Working with the campaignResponseDate Field

The behavior of the campaignResponseDate field varies depending on the type of operation you are performing:

- If you are using the **get** operation, the campaignResponseDate field maps to the field in the Campaign Response window labeled Date Created (shown in the red box in the following screenshot). The field indicates the date that the record was created.
- If you are using the **add** or **update** operation, the campaignResponseDate field maps to the fields in the Campaign Response window labeled New Response Date and New Response Time. These fields, shown in the blue box, represent a date and time that you choose when adding a record or a sublist line. If you do not choose a date and time, these fields default to the time that the record or sublist line was created.



Recipient	3 ABC Co.
Campaign	Trade Show
Event	[Default Event]
Date Created	7/8/2014
Channel	
New Response Date	7/9/2014
New Response Time	10:00 am
New Response Detail	Responded
Note	

RESPONSE DETAIL	DATE/TIME	AUTHOR	NOTES	
Responded	9/27/2013 8:45 am	John Smith	RSVP received for trade show.	Remove

For more-explicit examples of how this field behaves, see Example — Adding a Campaign Response Record and Example — Getting a Campaign Response Record.

See also Campaign Response Supported Operations for general guidance on the behavior of the add and update operations with this record.

Campaign Response Usage Notes

This section describes additional nuances of working with the campaign response record using SuiteTalk. See also Campaign Response Supported Operations for important details about the add and update operations.

Setting the Event Field for a Campaign Response

As of the 2012.2 endpoint, if you do not set the event field for a campaign response, the value of this field defaults to the [Default Event]. Note that endpoints prior to 2012.2 allow this field's value to remain unset.

Attaching a Campaign Response to a Contact

The following SOAP excerpt shows how to attach a campaign response to a contact. Be aware that you cannot attach campaign *records* to contacts, however you can attach campaign responses to contacts.

Note that you must set the **entity** field on CampaignResponse to reference the contact record.

```
<soapenv:Body>
   <platformMsgs:add ...
      <platformMsgs:record xsi:type="s0:CampaignResponse">
         <s0:entity internalId="8" type="contact" />
         <s0:leadSource internalId="10202" />
         <s0:response>_responded</s0:response>
      </platformMsgs:record>
   </platformMsgs:add>
</soapenv:Envelope>
```

Example — Adding a Campaign Response Record

The following example shows how to add a campaign response record. In this example, the campaignResponseDate is manually set.

C#

```
private void addCampaignResponse()
{

   CampaignResponse newCampaignResponse = new CampaignResponse();


   // Identify a campaign listed at Lists >
   // Marketing > Marketing Campaigns.

   RecordRef recordLeadSource = new RecordRef();
   recordLeadSource.type = RecordType.campaignResponse;
   recordLeadSource.internalId = "-5";
   newCampaignResponse.leadSource = recordLeadSource;


   // Identify an existing NetSuite entity record,
   // in this case a customer.

   RecordRef recordCustomer = new RecordRef();
   recordCustomer.type = RecordType.customer;
   recordCustomer.internalId = "1766";
   newCampaignResponse.entity = recordCustomer;
   // Identify the date and time that the customer responded,
   // using the campaignResponseDate field.

   DateTime lastWeek = new DateTime(2013, 9, 27, 8, 45, 00);
   newCampaignResponse.campaignResponseDate = lastWeek;
   newCampaignResponse.campaignResponseDateSpecified = true;


   // Optionally, specify a note and an external ID.

   newCampaignResponse.note = "note text";
   newCampaignResponse.externalId = "123456";


   // Execute operation.

   _service.add(newCampaignResponse);

}
```

SOAP Request

```
<soap:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.
org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Header> ...
    </soap:Header>
    <soap:Body>
        <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record externalId="123456" xsi:type="q1:CampaignResponse" xmlns:q1="urn:marketing_
2017_1.lists.webservices.netsuite.com">
                <q1:entity internalId="1766"/>
                <q1:leadSource internalId="-5"/>
                <q1:campaignResponseDate>2013-09-27T08:45:00</q1:campaignResponseDate>
                <q1:note>note text</q1:note>
            </record>
        </add>
    </soap:Body>
</soap:Envelope>
```

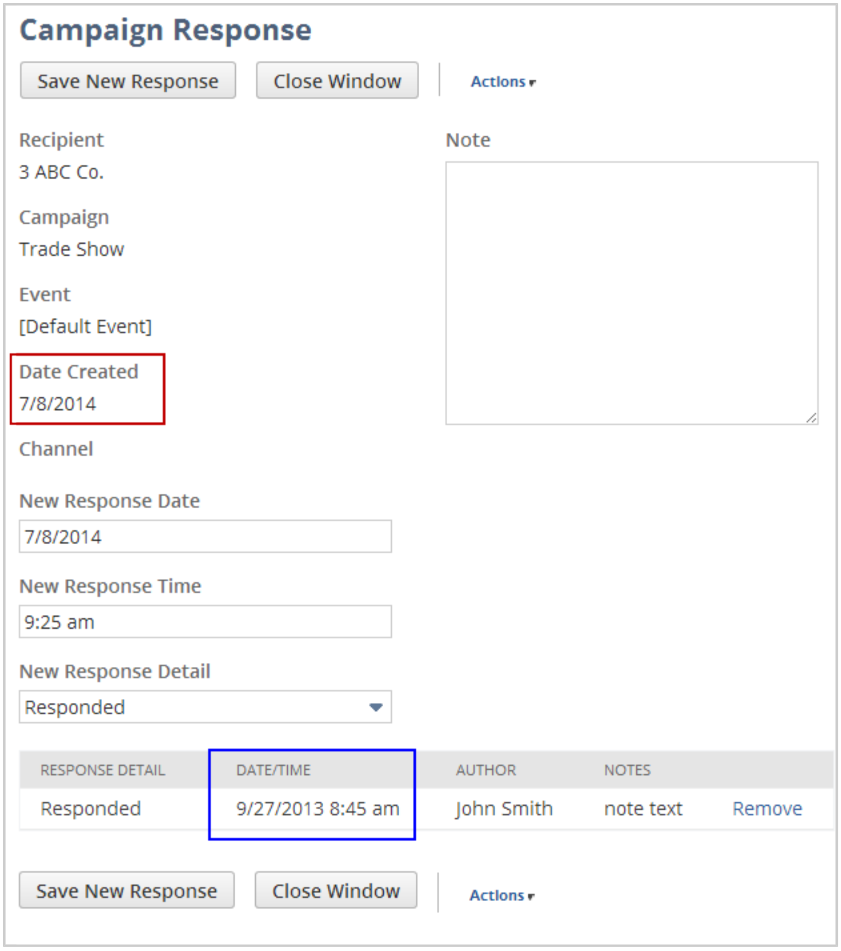
SOAP Response

```
<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> ...
    </soapenv:Header>
    <soapenv:Body>
        <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <writeResponse>
                <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platf
orm.webservices.netsuite.com"/>
                <baseRef xsi:type="platformCore:RecordRef" type="campaignResponse" externalId="
```

```
123456" internalId="303" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"
/>
            </writeResponse>
        </addResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

UI Result

The following screenshot shows the result of the SOAP request. The date in the red box is the date the SOAP request was sent. The date in the blue box is the date specified in the SOAP request.



Campaign Response

Recipient
3 ABC Co.

Campaign
Trade Show

Event
[Default Event]

Date Created
7/8/2014

Channel

New Response Date
7/8/2014

New Response Time
9:25 am

New Response Detail
Responded

Note

RESPONSE DETAIL	DATE/TIME	AUTHOR	NOTES	
Responded	9/27/2013 8:45 am	John Smith	note text	Remove

Example — Getting a Campaign Response Record

This sample shows how to get a campaign response record. Note that in this example, the campaignResponseDate value returned in the SOAP response is the date the record was created in NetSuite, not the date you might have specified representing the time of the entity's actual response. (The latter value is specified in the responseDate field of the responsesList. Note that you cannot modify the responsesList values, nor can you delete them through web services, though in some cases you can delete them through the UI.)

In response to a get operation, the campaignResponseDate field shows only a date, not a time. The time returned is always 00:00:00.000.

C#

```
private void getCampaignResponse()
{
   RecordRef recordCampaignResponse = new RecordRef();
   recordCampaignResponse.internalId = "303";
   recordCampaignResponse.type = RecordType.campaignResponse;
   recordCampaignResponse.typeSpecified = true;
   _service.get(recordCampaignResponse);
}
```

SOAP Request

```
<soap:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.
org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soap:Header> ...
    </soap:Header>
    <soap:Body>
        <get xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <baseRef type="campaignResponse" internalId="303" xsi:type="q1:RecordRef" xmlns:q1=
"urn:core_2017_1.platform.webservices.netsuite.com"/>
        </get>
    </soap:Body>
</soap: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> ...
    </soapenv:Header>
    <soapenv:Body>
        <getResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <readResponse>
                <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platf
orm.webservices.netsuite.com"/>
                <record xsi:type="listMkt:CampaignResponse" internalId="303" xmlns:listMkt="urn
:marketing_2017_1.lists.webservices.netsuite.com">
                    <listMkt:entity internalId="1766" xmlns:platformCore="urn:core_2017_1.platf
orm.webservices.netsuite.com">
                        <platformCore:name>3 ABC Co.</platformCore:name>
                    </listMkt:entity>
                    <listMkt:leadSource internalId="-5" xmlns:platformCore="urn:core_2017_1.pla
tform.webservices.netsuite.com">
                        <platformCore:name>557 Trade Show</platformCore:name>
                    </listMkt:leadSource>
                    <listMkt:campaignEvent internalId="13" xmlns:platformCore="urn:core_2017_1.
platform.webservices.netsuite.com">
                        <platformCore:name>[Default Event]</platformCore:name>
                    </listMkt:campaignEvent>
                    <listMkt:campaignResponseDate>2013-10-04T00:00:00.000-07:00</listMkt:campai
gnResponseDate>
```

```
                    <listMkt:response>_responded</listMkt:response>
                    <listMkt:responsesList>
                        <listMkt:responses>
                            <listMkt:response>Responded</listMkt:response>
                            <listMkt:responseDate>2013-09-27 08:45:00</listMkt:responseDate>
                            <listMkt:author>Smith, John</listMkt:author>
                            <listMkt:note>note text</listMkt:note>
                        </listMkt:responses>
                    </listMkt:responsesList>
                </record>
            </readResponse>
        </getResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Campaign Vertical

The campaign vertical record is defined in the listMkt (marketing) XSD.

Supported Operations

The following operations can be used with campaign vertical records:

add | addList | delete | deleteList | get | getAll | getList | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's campaign vertical reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Campaign Subscription

The campaign subscription record is defined in the listMkt (marketing) XSD.

Supported Operations

The following operations can be used with campaign subscription records:

add | addList | delete | deleteList | get | getAll | getList | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's campaign subscription reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Coupon Code

The coupon code record is defined in the listMkt (marketing) XSD.

Supported Operations

The following operations can be used with coupon code records:

add | addList | delete | deleteList | get | getDeleted | getList | search | searchMore | searchMoreWithId | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's coupon code reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Code Sample

The following sample shows how to add a coupon code through SuiteTalk. Note that to work with coupon codes, you must have a promotion with “single use” coupon codes (Promotion record > Codes subtab > Number of Uses > Single Use). Then you can start adding coupon codes.

Java

```
public void addCouponCode() throws Exception
   {
      this.login();
      CouponCode cc = new CouponCode();
      RecordRef promoRef = new RecordRef();
      promoRef.setInternalId("3");
      cc.setPromotion(promoRef);    // the promotion the coupon code is bound to

      cc.setExternalId("exCC12345");
      cc.setCode("CouponCode2012");

      RecordRef recRef = new RecordRef();
      recRef.setInternalId("15");
      cc.setRecipient(recRef);
      cc.setDateSent(Calendar.getInstance());
      _port.add(cc);
   }
```

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>
        <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record xsi:type="ns6:CouponCode" externalId="exCC12345" xmlns:ns6="urn:marketing_2
017_1.lists.webservices.netsuite.com">
                <ns6:promotion xsi:type="ns7:RecordRef" internalId="3" xmlns:ns7="urn:core_2017
_1.platform.webservices.netsuite.com"/>
                <ns6:code xsi:type="xsd:string">CouponCode2012</ns6:code>
                <ns6:recipient xsi:type="ns8:RecordRef" internalId="15" xmlns:ns8="urn:core_201
7_1.platform.webservices.netsuite.com"/>
                <ns6:dateSent xsi:type="xsd:dateTime">2012-08-10T08:13:27.370Z</ns6:dateSent>
            </record>
        </add>
    </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_08102012261640432423581449_323eee69f5203</pl
atformMsgs:nsId>
        </platformMsgs:documentInfo>
    </soapenv:Header>
    <soapenv:Body>
        <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <writeResponse>
                <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platf
orm.webservices.netsuite.com"/>
                <baseRef xsi:type="platformCore:RecordRef" type="couponCode" externalId="exCC12
345" internalId="3" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </writeResponse>
        </addResponse>
    </soapenv:Body>
</soapenv:Envelope>
```

Promotion Code

The promotion code record is defined in the listMkt (marketing) XSD.

Supported Operations

The following operations can be used with promotion code records:

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser's promotion code reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Warning: Line discount promotions are not compatible with 2009.2 and earlier endpoints. If you want to use these with web services, you must upgrade to the 2010.1 or later endpoint.

Code Sample

The following sample shows how to add a promotion code record to NetSuite.

Java

```
public void addPromotionCode()throws RemoteException
   {
      //This operation requires a valid session
      this.login(true);

      String readStr = "";
      PromotionCode pc = new PromotionCode();

      _console.writeLn("\nPlease enter promotion name: ");
      readStr = _console.readLn();
      pc.setCode(readStr);

      _console.writeLn("\nPlease enter description : ");
      readStr = _console.readLn();
      pc.setDescription(readStr);

      pc.setIsInactive(new Boolean(false));

      _console.writeLn("\nPlease enter the discount item internal ID: ");
      readStr = _console.readLn();
      RecordRef discount = new RecordRef();
      discount.setType(RecordType.discountItem);
      discount.setInternalId(readStr);
      pc.setDiscount(discount);

      //pc.setRate(String rate) //required if discount item not specified
      //pc.setDiscountType(Boolean discountType) //required if discount item not specified
      _console.writeLn("Do you want to apply code to first sale only or all sales?");

      while(true)
      {
         _console.writeLn("\nEnter 1 for First Sale Only, 2 for All Sales.");
         readStr = _console.readLn();
         PromotionCodeApplyDiscountTo applyDiscountTo = null;

         if(readStr.equalsIgnoreCase("1"))
         {
            applyDiscountTo = PromotionCodeApplyDiscountTo._firstSaleOnly;
            pc.setApplyDiscountTo(applyDiscountTo);
            break;
         }
         if(readStr.equalsIgnoreCase("2"))
         {
            applyDiscountTo = PromotionCodeApplyDiscountTo._allSales;
            pc.setApplyDiscountTo(applyDiscountTo);
            break;
         }
      }

      Calendar startDate = Calendar.getInstance();
      Calendar endDate = Calendar.getInstance();
```

```
       _console.writeLn("Enter promotion code starting day.");
       readStr = _console.readLn();
       int day = Integer.parseInt(readStr);

       _console.writeLn("Enter promotion code starting month (1-12).");
       readStr = _console.readLn();
       int month = Integer.parseInt(readStr);

       //January=0
       startDate.set(2009, month-1, day);
       pc.setStartDate(startDate);

       _console.writeLn("Enter promotion code ending day.");
       readStr = _console.readLn();
       day = Integer.parseInt(readStr);

       _console.writeLn("Enter promotion code ending month (1-12).");
       readStr = _console.readLn();
       month = Integer.parseInt(readStr);

       endDate.set(2009, month-1, day);
                pc.setEndDate(endDate);

       pc.setIsPublic(new Boolean(true));
       pc.setExcludeItems(new Boolean(false));

       WriteResponse writeRes = _port.add(pc);
       boolean success = writeRes.getStatus().isIsSuccess();
       if (success)
       {
          _console.writeLn("\nPromotion Code created successfully.");
          RecordRef _ref = (RecordRef)(writeRes.getBaseRef());
          _console.writeLn("Internal ID: " + _ref.getInternalId());
       }
       else
       {
          _console.error(getStatusDetails(writeRes.getStatus()));
       }
    }
```

SOAP Request

```
<soapenv:Body>
      <platformMsgs:add
          xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
          xmlns:listMktTyp="urn:types.marketing_2017_1.lists.webservices.netsuite.com"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:platformMsgs="urn:messages_2017_1.platform.webservices.netsuite.com"
          xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"
          xmlns:xs="http://www.w3.org/2001/XMLSchema"
          xmlns:s0="urn:marketing_2017_1.lists.webservices.netsuite.com"
          xmlns:s1="urn:sales_2017_1.transactions.webservices.netsuite.com">
         <platformMsgs:record xsi:type="s0:PromotionCode">
            <s0:code>Any Promo Code</s0:code>
            <s0:isInactive>false</s0:isInactive>
            <s0:discount internalId="28" />
            <s0:applyDiscountTo>_firstSaleOnly</s0:applyDiscountTo>
            <s0:startDate>2009-01-09T23:23:38.787Z</s0:startDate>
            <s0:endDate>2009-01-16T18:23:52.000Z</s0:endDate>
            <s0:isPublic>true</s0:isPublic>
            <s0:excludeItems>false</s0:excludeItems>
         </platformMsgs:record>
      </platformMsgs:add>
   </soapenv:Body>
</soapenv:Envelope>
```

File Cabinet

The following file cabinet records are supported in SuiteTalk:

- File
- Folder

File

The File record type is defined in the docFileCab (fileCabinet) XSD.

For details on working with this record in web services, see the following sections:

- File Record Supported Operations
- File Record Field Definitions
- File Record Code Samples

For help working with this record in the UI, see the help topic Working with the File Cabinet.

File Record Supported Operations

The following operations can be used with the file record.

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

When you **add** a file, be aware of the following:

- You must always specify a value for the name field. If you do not specify a file name, the system returns an error. This behavior differs from the UI, where the record’s name field is automatically set to the name of the file.
- You must include a value for the attachFrom enumeration. Use _computer if you are uploading a local file. Use _web if you are uploading a file from the Internet.
- The maximum file size is 10 MB.

If you want to retrieve data about a file record and the file that it represents, be aware of the following:

- The only way to retrieve the content of an uploaded file is by using a single **get** operation. When using either the **getList** or **search** operation, NetSuite returns details about the file record, but not the content of the file that was uploaded.
- When the content of a file is returned, it is in base64Binary format.
- During a **search**, the bodyFieldsOnly search preference has no bearing on file content.

File Record Field Definitions

The following table summarizes the required body fields on the file record type.

Field Name	Type
attachFrom	Enumeration
content	base64Binary
folder	RecordRef
name	String

For examples of how to use these fields when adding a file record, see File Record Code Samples.

Note: For more details on this record, see the Schema Browser's file reference page. The Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

File Record Code Samples

The following examples show how to create a file record by uploading a local file.

Java

```
public void testAddFile() throws Exception
{
   c.setCredentials(CRED_DEV_MSTRWLF);
   c.setHttpPort(1120);

   c.login();


   // identify and load local file

   com.netsuite.webservices.documents.filecabinet.File file = new File();
   java.io.File file = (new java.io.File("/webdev/NetLedger/test/resource/info/test.csv"));
   f.setContent( FileUtils.readFileToByteArray(file) );
   f.setAttachFrom(FileAttachFrom._computer);
   f.setName(file.getName());
   f.setExternalId("101A");


   // set folder

   f.setFolder(createRR("-4"));


   // call NetSuite service
```

```
    c.addRecord(f);
}
```

C#

```
{
NetSuite.com.netsuite.webservices.File myFile = new File();

    myFile.name = "test";
    myFile.attachFrom = FileAttachFrom._computer;
    myFile.attachFromSpecified = true;


    // identify local file

    string fileName = "test.csv";


    // load file

    System.IO.FileStream inFile;
    byte[] data;

    inFile = new System.IO.FileStream(fileName, System.IO.FileMode.Open, System.IO.FileAccess.Re
ad);
    data = new Byte[inFile.Length];
    long bytesRead = inFile.Read(data, 0, (int)inFile.Length);
    inFile.Close();


    // set Folder

    RecordRef folderRef = new RecordRef();
    folderRef.internalId = "17";
    myFile.folder = folderRef;


    // identify content

    myFile.content = data;


    // call NetSuite service

    _service.add(myFile);


}
```

SOAP Request

```
<soap:Body>
```

```
   <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <record xsi:type="q1:File" xmlns:q1="urn:filecabinet_2017_1.documents.webservices.netsuit
e.com">
         <q1:name>test</q1:name>
         <q1:attachFrom>_computer</q1:attachFrom>
         <q1:content>RXh0ZXJuYWwgSUQsTmFtZQ0KMTAzLHNjYXJsZXQNCjEwMixkYXduDQoxMDQsc3VuDQo=</q1:c
ontent>
         <q1:folder internalId="17"/>
      </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="file" internalId="6914" xmlns:platfor
mCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
   </addResponse>
</soapenv:Body>
```

Folder

The folder record is defined in docFileCab (fileCabinet) XSD.

Supported Operations

The following operations can be used to manipulate the folder record.

add | addList | delete | deleteList | get | getDeleted | getList | getSavedSearch | getSelectValue | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this record. For details, see the Schema Browser’s folder reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Subrecords

This topic lists some of the subrecord types that are supported in web services:

- Address
- Inventory Detail
- Landed Cost

For general details about working with subrecords in web services, see the help topic Subrecords.

Address

The address subrecord is defined in the Common XSD. This subrecord was exposed to web services in the 2014.2 endpoint.

For information about address form customization, see the help topic Customizing Address Forms.

Supported Operations

add | addList | delete | deleteList | get | getList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Operations Must Be in Parent Context

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

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this subrecord. For details, see the Schema Browser's address reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

Address Preferences

The following preferences apply to the address subrecord, and generally to all 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.

Sample Code

The following sections show examples of how to interact with the address subrecord.

Adding a New Record with the Address Specified

The following example illustrates adding a new record with the billing and shipping addresses specified.

SOAP Request

```
<soapenv:Body>
         <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">

            <record xsi:type="ns6:Customer" xmlns:ns6="urn:relationships_2017_1.lists.webservic
es.netsuite.com">
               <ns6:isPerson xsi:type="xsd:boolean">true</ns6:isPerson>

               <ns6:firstName xsi:type="xsd:string">Maria</ns6:firstName>

               <ns6:lastName xsi:type="xsd:string">Magnusson</ns6:lastName>

               <ns6:subsidiary internalId="3" xsi:type="ns7:RecordRef" xmlns:ns7="urn:core_2017
_1.platform.webservices.netsuite.com"/>
               <ns6:addressbookList replaceAll="false" xsi:type="ns6:CustomerAddressbookList">

                  <ns6:addressbook xsi:type="ns6:CustomerAddressbook">

                     <ns6:label xsi:type="xsd:string">Billing address</ns6:label>
                     <ns6:defaultShipping xsi:type="xsd:boolean">false</ns6:defaultShipping>

                     <ns6:addressbookAddress xsi:type="ns8:Address" xmlns:ns8="urn:common_2017_
1.platform.webservices.netsuite.com">

                        <ns8:country xsi:type="ns9:Country" xmlns:ns9="urn:types.common_2017_1.
platform.webservices.netsuite.com">_unitedStates</ns8:country>

                        <ns8:addressee xsi:type="xsd:string">Maria Magnusson</ns8:addressee>

                        <ns8:addr1 xsi:type="xsd:string">Hill Crescent 254</ns8:addr1>

                        <ns8:city xsi:type="xsd:string">Beverly Hills</ns8:city>

                        <ns8:state xsi:type="xsd:string">CA</ns8:state>

                        <ns8:zip xsi:type="xsd:string">90210</ns8:zip>

                     </ns6:addressbookAddress>
```

```
                    </ns6:addressbook>

                    <ns6:addressbook xsi:type="ns6:CustomerAddressbook">

                       <ns6:label xsi:type="xsd:string">Shipping address</ns6:label>
                       <ns6:defaultShipping xsi:type="xsd:boolean">true</ns6:defaultShipping>

                       <ns6:addressbookAddress xsi:type="ns10:Address" xmlns:ns10="urn:common_201
7_1.platform.webservices.netsuite.com">
                          <ns10:country xsi:type="ns11:Country" xmlns:ns11="urn:types.common_2017
_1.platform.webservices.netsuite.com">_unitedStates</ns10:country>

                          <ns10:addressee xsi:type="xsd:string">Maria Magnusson</ns10:addressee>

                          <ns10:addr1 xsi:type="xsd:string">Broadway 75</ns10:addr1>

                          <ns10:city xsi:type="xsd:string">New York City</ns10:city>

                          <ns10:state xsi:type="xsd:string">NY</ns10:state>

                          <ns10:zip xsi:type="xsd:string">10005</ns10:zip>

                       </ns6:addressbookAddress>

                    </ns6:addressbook>
                 </ns6:addressbookList>
              </record>
           </add>
        </soapenv: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.platfo
rm.webservices.netsuite.com"/>
                 <baseRef internalId="630" type="customer" xsi:type="platformCore:RecordRef" xmln
s:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
              </writeResponse>
           </addResponse>
        </soapenv:Body>
```

Updating a Record with a New Address

In this example, a customer’s address is replaced with a new address. To update an address, you have to use a keyed sublist. For information about working with keyed sublists, see the help topic Updating Sublists in Web Services.

SOAP Request

```
        <soapenv:Body>
```

```
         <update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record internalId="831" xsi:type="ns6:Customer" xmlns:ns6="urn:relationships_2017_
1.lists.webservices.netsuite.com">
               <ns6:addressbookList replaceAll="true" xsi:type="ns6:CustomerAddressbookList">

                  <ns6:addressbook xsi:type="ns6:CustomerAddressbook">
                     <ns6:defaultShipping xsi:type="xsd:boolean">true</ns6:defaultShipping>

                     <ns6:defaultBilling xsi:type="xsd:boolean">true</ns6:defaultBilling>

                     <ns6:label xsi:type="xsd:string">Main address</ns6:label>
                     <ns6:addressbookAddress xsi:type="ns7:Address" xmlns:ns7="urn:common_2017_
1.platform.webservices.netsuite.com">
                        <ns7:country xsi:type="ns8:Country" xmlns:ns8="urn:types.common_2017_1.
platform.webservices.netsuite.com">_unitedStates</ns7:country>
                        <ns7:addressee xsi:type="xsd:string">Harriet Thomas</ns7:addressee>

                        <ns7:addr1 xsi:type="xsd:string">Broadway 65</ns7:addr1>
                        <ns7:city xsi:type="xsd:string">New York</ns7:city>
                        <ns7:zip xsi:type="xsd:string">10008</ns7:zip>
                     </ns6:addressbookAddress>
                  </ns6:addressbook>
               </ns6:addressbookList>
            </record>
         </update>
      </soapenv:Body>
```

SOAP Response

```
      <soapenv:Body>
         <updateResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <writeResponse>
               <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
               <baseRef internalId="831" type="customer" xsi:type="platformCore:RecordRef" xmln
s:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </writeResponse>
         </updateResponse>
      </soapenv:Body>
```

Searching for Addresses

The following example illustrates how to search for the address details of records. In this example, a city is specified as the search term. The response lists all records with the specified city in the address.

SOAP Request

```
      <soapenv:Body>
         <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <searchRecord xsi:type="ns6:AddressSearchBasic" xmlns:ns6="urn:common_2017_1.platfo
rm.webservices.netsuite.com">
               <ns6:city operator="is" xsi:type="ns7:SearchStringField" xmlns:ns7="urn:core_201
7_1.platform.webservices.netsuite.com">
```

```
                  <ns7:searchValue xsi:type="xsd:string">Edinburgh</ns7:searchValue>
               </ns6:city>
            </searchRecord>
         </search>
      </soapenv:Body>
```

SOAP Response

```
      <soapenv:Body>
         <searchResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <platformCore:searchResult xmlns:platformCore="urn:core_2017_1.platform.webservices
.netsuite.com">
               <platformCore:status isSuccess="true"/>
               <platformCore:totalRecords>1</platformCore:totalRecords>
               <platformCore:pageSize>1000</platformCore:pageSize>
               <platformCore:totalPages>1</platformCore:totalPages>
               <platformCore:pageIndex>1</platformCore:pageIndex>
               <platformCore:searchId>WEBSERVICES_3604360_1030201718380211442049908681_fb97b295
8d9</platformCore:searchId>
               <platformCore:recordList>
                  <platformCore:record xsi:type="platformCommon:Address" xmlns:platformCommon="
urn:common_2017_1.platform.webservices.netsuite.com">
                     <platformCommon:internalId>306</platformCommon:internalId>
                     <platformCommon:country>_unitedKingdomGB</platformCommon:country>
                     <platformCommon:addressee>Hella Bigsmile</platformCommon:addressee>
                     <platformCommon:addr1>Montgomery Street 5</platformCommon:addr1>
                     <platformCommon:city>Edinburgh</platformCommon:city>
                     <platformCommon:state>E Lothian</platformCommon:state>
                     <platformCommon:addrText>Hella Bigsmile&lt;br&gt;Montgomery Street 5&lt;br
&gt;Edinburgh&lt;br&gt;E Lothian&lt;br&gt;United Kingdom</platformCommon:addrText>
                     <platformCommon:override>false</platformCommon:override>
                  </platformCore:record>
               </platformCore:recordList>
            </platformCore:searchResult>
         </searchResponse>
      </soapenv:Body>
```

Inventory Detail

The inventory detail subrecord is available when the Advanced Bin / Numbered Inventory Management feature is enabled.

- This type of record stores values relating to bin numbers and/or serial/lot numbers for items, including line items on transactions.
- This data includes the quantity on hand and quantity available per bin number, or per serial/lot number, or when both are in use, per bin number/serial lot number combination.
- In the UI, the inventory detail subrecord displays as a popup when you click the Inventory Detail body field or sublist field.
- For more details, see the help topic Advanced Bin / Numbered Inventory Management.

The inventory detail subrecord is defined in the Common XSD.

Important: The web services code used to access bin number and serial/lot number data for items varies according to whether the Advanced Bin / Numbered Inventory Management feature is enabled. If this feature is not enabled, this data is available directly from the item, and you receive an error if you attempt to access it through the inventory detail record. If this feature is enabled, this data must be accessed through the inventory detail record. See Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled.

Supported Operations

add | addList | delete | deleteList | get | getList | search | searchMore | searchNext | update | updateList | upsert | upsertList

Note: You can also use the asynchronous equivalents of web services list operations. For information about asynchronous operations, see the help topic Web Services Asynchronous Operations. For more information about request processing, see the help topic Synchronous Versus Asynchronous Request Processing.

Operations Must Be in Parent Context

Any add, get, update, delete, or search operation on an inventory detail 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 an item on a purchase order transaction, you must update the purchase order record itself; you cannot do an independent update of the inventory detail object. And this same limitation applies to all operations on an inventory detail subrecord.

Field Definitions

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this subrecord. For details, see the Schema Browser’s inventory detail reference page

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Usage Notes

See the following sections for more details about this subrecord:

- InventoryAssignmentList
- Inventory Detail Preferences

InventoryAssignmentList

In the schema, InventoryDetail includes a sublist called InventoryAssignmentList, with sublist fields that include the following: internalId, issueInventoryNumber, receiptnventoryNumber, binNumber, toBinNumber, quantity, expirationDate, and quantityAvailable.

Inventory Detail Preferences

The following preferences apply to inventory detail (and generally to all 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. For inventory detail, the default for the replaceAll attribute is false.

Sample Code

The following sections show a few examples of how to interact with the inventory details subrecord.

Setting Inventory Details on an Adjustment

The following examples illustrate setting inventory detail values on an inventory adjustment transaction.

Java

```
InventoryAdjustment ia = new InventoryAdjustment();
ia.setSubsidiary(new RecordRef(null,"1",null,null));
ia.setAccount(new RecordRef(null,"1",null,null));

InventoryAdjustmentInventory item = new InventoryAdjustmentInventory();
item.setItem(new RecordRef(null, i.getInternalId(), null, null));
item.setLocation(new RecordRef(null,"1",null,null));
item.setAdjustQtyBy(new Double(2));

InventoryAssignment assign = new InventoryAssignment();
assign.setReceiptInventoryNumber("Grape19816143,Melon12289447");
assign.setQuantity(new Double(2));

InventoryDetail id = new InventoryDetail();
id.setInventoryAssignmentList(new InventoryAssignmentList(new InventoryAssignment[] {assign},tr
ue));
item.setInventoryDetail(id);

ia.setInventoryList(new InventoryAdjustmentInventoryList(new InventoryAdjustmentInventory[] {it
em},true));

sessMgr.getPort().add(ia);
```

SOAP Request

```
<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:Body>
         <add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record externalId="Strawberry10424664" xsi:type="ns6:InventoryAdjustment" xmlns:ns
```

```
6="urn:inventory_2017_1.transactions.webservices.netsuite.com">
               <ns6:subsidiary internalId="1" xsi:type="ns7:RecordRef" xmlns:ns7="urn:core_2017
_1.platform.webservices.netsuite.com"/>
               <ns6:account internalId="1" xsi:type="ns8:RecordRef" xmlns:ns8="urn:core_2017_1.
platform.webservices.netsuite.com"/>
               <ns6:inventoryList replaceAll="true" xsi:type="ns6:InventoryAdjustmentInventoryL
ist">
                  <ns6:inventory xsi:type="ns6:InventoryAdjustmentInventory">
                     <ns6:item internalId="169" xsi:type="ns9:RecordRef" xmlns:ns9="urn:core_20
17_1.platform.webservices.netsuite.com"/>
                     <ns6:inventoryDetail xsi:type="ns10:InventoryDetail" xmlns:ns10="urn:commo
n_2017_1.platform.webservices.netsuite.com">
                        <ns10:inventoryAssignmentList replaceAll="true" xsi:type="ns10:Inventor
yAssignmentList">
                           <ns10:inventoryAssignment xsi:type="ns10:InventoryAssignment">
                              <ns10:receiptInventoryNumber xsi:type="xsd:string">Grape19816143,
Melon12289447</ns10:receiptInventoryNumber>
                              <ns10:quantity xsi:type="xsd:double">2.0</ns10:quantity>
                           </ns10:inventoryAssignment>
                        </ns10:inventoryAssignmentList>
                     </ns6:inventoryDetail>
                     <ns6:location internalId="1" xsi:type="ns11:RecordRef" xmlns:ns11="urn:cor
e_2017_1.platform.webservices.netsuite.com"/>
                     <ns6:adjustQtyBy xsi:type="xsd:double">2.0</ns6:adjustQtyBy>
                  </ns6:inventory>
               </ns6:inventoryList>
            </record>
         </add>
      </soapenv:Body>
   </soapenv:Envelope>
```

SOAP Response

```
<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.webservice
s.netsuite.com">
            <platformMsgs:nsId>WEBSERVICES_1247336_092920111272224852700968262_b9abc599312e</pl
atformMsgs:nsId>
         </platformMsgs:documentInfo>
      </soapenv:Header>
      <soapenv:Body>
         <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <writeResponse>
               <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platfo
rm.webservices.netsuite.com"/>
               <baseRef internalId="76" externalId="Strawberry10424664" type="inventoryAdjustme
nt" xsi:type="platformCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.
netsuite.com"/>
            </writeResponse>
         </addResponse>
      </soapenv:Body>
   </soapenv:Envelope>
```

Searching by Inventory or Bin Number

The following examples illustrate how to search using inventory number or bin number.

With the first approach shown below, you search using the internal ID of the inventory number. (An inventory number is a serial number or a lot number.) In response, the system looks for item records that reference this number. For each occurrence found, the system returns the data available in the Inventory Detail pop-up. Note that these details may be associated with a variety of other records, such as sales orders, inventory adjustments, and so on. This data includes the serial/lot number, the quantity and, if applicable, the bin number.

SOAP Request for Search by Inventory Number

```
<search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <searchRecord xsi:type="ns7:ItemSearchAdvanced" xmlns:ns7="urn:accounting_2017_1.li
sts.webservices.netsuite.com">
               <ns7:criteria xsi:type="ns7:ItemSearch">
                  <ns7:inventoryDetailJoin xsi:type="ns8:InventoryDetailSearchBasic" xmlns:ns8=
"urn:common_2017_1.platform.webservices.netsuite.com">
                     <ns8:inventoryNumber operator="anyOf" xsi:type="ns9:SearchMultiSelectField
" xmlns:ns9="urn:core_2017_1.platform.webservices.netsuite.com">
                        <ns9:searchValue internalId="143" type="inventoryNumber" xsi:type="ns9:
RecordRef"/>
                     </ns8:inventoryNumber>
                  </ns7:inventoryDetailJoin>
               </ns7:criteria>
               <ns7:columns xsi:type="ns7:ItemSearchRow">
                  <ns7:inventoryDetailJoin xsi:type="ns10:InventoryDetailSearchRowBasic" xmlns:
ns10="urn:common_2017_1.platform.webservices.netsuite.com">
                     <ns10:binNumber xsi:type="ns11:SearchColumnSelectField" xmlns:ns11="urn:co
re_2017_1.platform.webservices.netsuite.com"/>
                     <ns10:internalId xsi:type="ns12:SearchColumnSelectField" xmlns:ns12="urn:c
ore_2017_1.platform.webservices.netsuite.com"/>
                     <ns10:inventoryNumber xsi:type="ns13:SearchColumnSelectField" xmlns:ns13="
urn:core_2017_1.platform.webservices.netsuite.com"/>
                     <ns10:quantity xsi:type="ns14:SearchColumnDoubleField" xmlns:ns14="urn:cor
e_2017_1.platform.webservices.netsuite.com"/>
                  </ns7:inventoryDetailJoin>
               </ns7:columns>
            </searchRecord>
         </search>
```

The following example is similar to the preceding one. However, in this example, you search using the internal ID of the bin number. Similar to the previous example, this search looks for item records, this time looking for those whose inventory details reference the bin number identified in your search. Again, these details may be associated with a variety of records. The data returned includes the serial/lot number, the quantity and the bin number.

SOAP Request for Search by Bin Number

```
<search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <searchRecord xsi:type="ns7:ItemSearchAdvanced" xmlns:ns7="urn:accounting_2017_1.li
sts.webservices.netsuite.com">
```

```
                <ns7:criteria xsi:type="ns7:ItemSearch">
                   <ns7:inventoryDetailJoin xsi:type="ns8:InventoryDetailSearchBasic" xmlns:ns8=
"urn:common_2017_1.platform.webservices.netsuite.com">
                      <ns8:binNumber operator="anyOf" xsi:type="ns9:SearchMultiSelectField" xmln
s:ns9="urn:core_2017_1.platform.webservices.netsuite.com">
                         <ns9:searchValue internalId="1" type="bin" xsi:type="ns9:RecordRef"/>
                      </ns8:binNumber>
                   </ns7:inventoryDetailJoin>
                </ns7:criteria>
                <ns7:columns xsi:type="ns7:ItemSearchRow">
                   <ns7:inventoryDetailJoin xsi:type="ns10:InventoryDetailSearchRowBasic" xmlns:
ns10="urn:common_2017_1.platform.webservices.netsuite.com">
                      <ns10:binNumber xsi:type="ns11:SearchColumnSelectField" xmlns:ns11="urn:co
re_2017_1.platform.webservices.netsuite.com"/>
                      <ns10:internalId xsi:type="ns12:SearchColumnSelectField" xmlns:ns12="urn:c
ore_2017_1.platform.webservices.netsuite.com"/>
                      <ns10:inventoryNumber xsi:type="ns13:SearchColumnSelectField" xmlns:ns13="
urn:core_2017_1.platform.webservices.netsuite.com"/>
                      <ns10:quantity xsi:type="ns14:SearchColumnDoubleField" xmlns:ns14="urn:cor
e_2017_1.platform.webservices.netsuite.com"/>
                   </ns7:inventoryDetailJoin>
                </ns7:columns>
             </searchRecord>
          </search>
```

Updating Web Services Code When Advanced Bin / Numbered Inventory Management is Enabled

The web services code used to access bin number and serial/lot number data for items, including transaction line items, varies according to whether the Advanced Bin / Numbered Inventory Management feature is enabled. If this feature is not enabled, this data is available directly from the item. If this feature is enabled, this data must be accessed through the inventory detail subrecord, which was introduced in the 2011.2 endpoint.

If this feature is enabled in your account, you need to do the following to avoid unexpected results or errors in your web services code.

1. Update to the 2011.2 endpoint or later to have access to the inventory detail subrecord and its bin and numbered inventory fields.
2. Review the Inventory Detail subrecord to get an understanding of how it is structured in the schema.
3. Review the list of Records that May Include Inventory Detail Data.
4. Review any existing code for these records that accesses bin or numbered inventory fields, and update this code to use the inventory detail subrecord. See Sample Code Changes after Enabling Advanced Bin / Numbered Inventory Management.

Sample Code Changes after Enabling Advanced Bin / Numbered Inventory Management

The following code samples illustrate the difference between:

- Setting a serial number value for an item in an inventory adjustment transaction without Advanced Bin / Numbered Inventory Management, and
- Setting inventory detail data, including the serial number, with the Advanced Bin / Numbered Inventory Management feature enabled.

When the feature is enabled, code like the bolded, red text on the left must change to code like the bolded, red text on the right.

WITHOUT Advanced Bin/Numbered Inventory Management

```
InventoryAdjustment ia = new
InventoryAdjustment();
ia.setSubsidiary(new RecordRef(null,"1",null,null));
ia.setAccount(new RecordRef(null,"1",null,null));
InventoryAdjustmentInventory item = new
InventoryAdjustmentInventory();
item.setItem(new RecordRef(null, i.getInternalId(),
null, null));
item.setLocation(new RecordRef(null,"1",null,null));
item.setAdjustQtyBy(new Double(2));
item.setSerialNumbers("Grape16324299(2)");
ia.setInventoryList(new
InventoryAdjustmentInventoryList(new
InventoryAdjustmentInventory[] {item},true));
sessMgr.getPort().add(ia);
```

With Advanced Bin/Numbered Inventory Management ENABLED

```
InventoryAdjustment ia = new
InventoryAdjustment();
ia.setSubsidiary(new RecordRef(null,"1",null,null));
ia.setAccount(new RecordRef(null,"1",null,null));
InventoryAdjustmentInventory item = new
InventoryAdjustmentInventory();
item.setItem(new RecordRef(null, i.getInternalId(),
null, null));
item.setLocation(new RecordRef(null,"1",null,null));
item.setAdjustQtyBy(new Double(2));
InventoryAssignment assign = new
InventoryAssignment();
assign.setReceiptInventoryNumber("
Grape19816143,Melon12289447");
assign.setQuantity(new Double(2));
InventoryDetail id = new InventoryDetail();
id.setInventoryAssignmentList(new
InventoryAssignmentList(new
InventoryAssignment[] {assign},true));
item.setInventoryDetail(id);
ia.setInventoryList(new
InventoryAdjustmentInventoryList(new
InventoryAdjustmentInventory[] {item},true));
sessMgr.getPort().add(ia);
```

Records that May Include Inventory Detail Data

Review the following lists to understand which code you may need to modify after enabling the Advanced Bin / Numbered Inventory Management feature.

Items Affected by Advanced Bin/Numbered Inventory Management

The following types of items include inventory detail data when the Advanced Bin / Numbered Inventory Management feature is enabled:

- Assembly Item (BOM Item) (when Use Bins is set to True)
- Lot Numbered Assembly Item
- Serialized Assembly Item
- Inventory Item (when Use Bins is set to True)
- Lot Numbered Inventory Item
- Serialized Inventory Item

Transactions Affected by Advanced Bin / Numbered Inventory Management

The following types of transactions include inventory detail data for each line item when the Advanced Bin / Numbered Inventory Management feature is enabled:

- Assembly Build
- Assembly Unbuild
- Bin Putaway Worksheet
- Bin Transfer
- Cash Refund
- Cash Sale
- Check
- Credit Memo
- Estimate/Quote
- Inventory Adjustment
- Inventory Transfer
- Invoice
- Item Fulfillment
- Item Receipt
- Purchase Order
- Return Authorization
- Sales Order
- Transfer Order
- Vendor Bill
- Vendor Credit
- Vendor Return Authorization
- Work Order

Landed Cost

The landed cost subrecord is defined in the Common XSD.

The SuiteTalk Schema Browser includes definitions for all body fields, sublist fields, search filters, and search joins available to this subrecord. For details, see the Schema Browser's landed cost reference page.

Note: For information on using the SuiteTalk Schema Browser, see SuiteTalk Schema Browser.

Searches

This topic lists some of the search types that are supported in SuiteTalk:

- Accounting Transaction Search
- Item Search
- Transaction Search

Accounting Transaction Search

If your account has the Multi-Book Accounting feature enabled, you can use web services to search for transactions using accounting book as a search filter or a search column. To execute this type of search, you use the Multi-Book Accounting Transaction Search type.

In the user interface, you can view this search type by navigating to Reports > New Search and clicking Multi-Book Accounting Transaction.

For more information about this search type, refer to the SuiteTalk Schema Browser’s AccountingTransactionSearch reference page.

Item Search

Most item record types use the ItemSearch complex type for search. The **basic** element in ItemSearch references ItemSearchBasic, which lists all of the filter fields available when searching items.

The ItemSearch record also lists all search joins available in an item search. For details, see the Schema Browser’s ItemSearch reference page. The ItemSearch complex type is defined in the lists accounting XSD.

Be aware that the search filter fields available vary depending on the item type you are searching. For example, not all of the search filter fields defined in ItemSearchBasic exist on the Subtotal Item record.

Important: By default, only a record's body fields are returned on a search. If you want to return the information specified on a record's sublist, you must set the **bodyFieldsOnly** element of the SearchPreferences type to **false**. For general information on searching in web services, see the help topic search.

For more details on this search type, see the following topics:

- Item Search Usage Notes
- Item Search Code Sample

Item Search Usage Notes

See the following for more details about the Item Search complex type:

- When Multiple Price Feature Is Enabled
- The ItemID Element
- Description Field for Item Group

When Multiple Price Feature Is Enabled

In regard to the ItemSearchAdvanced complex type, note the following: When the Multiple Prices feature is enabled, the otherPrices field returns all existing prices other than the basePrice, including alternate prices and online price. This behavior matches that of the Other Prices search results field in UI advanced item searches.

The ItemID Element

Searching by ItemID, which is part of ItemSearchBasic, is the same as using the “Name (Internal)” filter that is available in the UI's advanced-search form. “Name (Internal)” searches the field on the item record labeled Item Name/Number.

Note that ItemID behave this way regardless of which endpoint is being used. This information supersedes the guidance available in deprecated versions of the Schema Browser.

Description Field for Item Group

It is not possible to add the description field of the Item Group as the Search column in the Item group advanced search, as it is not defined in the ItemSearchRowBasic ComplexType.

Item Search Code Sample

The following example shows how to search for all inventory items that use the word “cashmere” in their internal name field.

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.searchValue = searchStringArray;
   myItemSearchBasic.type = myEnum;

   SearchStringField myName = new SearchStringField();
   myName.@operator = SearchStringFieldOperator.contains;
   myName.operatorSpecified = true;
   String myNameValue = "cashmere";
   myName.searchValue = myNameValue;
   myItemSearchBasic.displayName = myName;

   ItemSearch myItemSearch = new ItemSearch();
   myItemSearch.basic = myItemSearchBasic;

   SearchResult searchResult = _service.search(myItemSearch);
}
```

SOAP Request

```
<soap:Body>
   <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <searchRecord xsi:type="q1:ItemSearch" xmlns:q1="urn:accounting_2017_1.lists.webservices.
netsuite.com">
       <q1:basic>
            <displayName operator="contains" xmlns="urn:common_2017_1.platform.webservices.nets
uite.com">
               <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">cashmere<
/searchValue>
            </displayName>
            <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>
      </searchRecord>
   </search>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
   <searchResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <platformCore:searchResult xmlns:platformCore="urn:core_2017_1.platform.webservices.netsu
ite.com">
         <platformCore:status isSuccess="true"/>
         <platformCore:totalRecords>2</platformCore:totalRecords>
       <platformCore:pageSize>1000</platformCore:pageSize>
       <platformCore:totalPages>1</platformCore:totalPages>
       <platformCore:pageIndex>1</platformCore:pageIndex>
       <platformCore:searchId>WEBSERVICES_000071_062020152867517261670438420_ab84bfa4e9f1b</pla
tformCore:searchId>
       <platformCore:recordList>
          <platformCore:record xsi:type="listAcct:InventoryItem" externalId="2000A" internalId=
"99" xmlns:listAcct="urn:accounting_2017_1.lists.webservices.netsuite.com">
               <listAcct:createdDate>2015-06-20T02:49:21.000-07:00</listAcct:createdDate>
               <listAcct:lastModifiedDate>2015-06-20T03:51:12.000-07:00</listAcct:lastModifiedD
ate>
               <listAcct:copyDescription>false</listAcct:copyDescription>
             <listAcct:cogsAccount internalId="55">
                  <platformCore:name>5000 Purchases</platformCore:name>
               </listAcct:cogsAccount>
               <listAcct:includeChildren>false</listAcct:includeChildren>
               <listAcct:incomeAccount internalId="54">
                <platformCore:name>4000 Sales</platformCore:name>
               </listAcct:incomeAccount>
               <listAcct:taxSchedule internalId="1">
```

```
                <platformCore:name>S1</platformCore:name>
             </listAcct:taxSchedule>
             <listAcct:deferRevRec>false</listAcct:deferRevRec>
             <listAcct:assetAccount internalId="10">
                <platformCore:name>1200 Inventory</platformCore:name>
             </listAcct:assetAccount>
                  ...
             <listAcct:outOfStockBehavior>_default</listAcct:outOfStockBehavior>
             <listAcct:itemId>Cashmere Sweater</listAcct:itemId>
             <listAcct:displayName>Cashmere Sweater</listAcct:displayName>
             <listAcct:isOnline>false</listAcct:isOnline>
             <listAcct:offerSupport>false</listAcct:offerSupport>
             <listAcct:isInactive>false</listAcct:isInactive>
             <listAcct:availableToPartners>false</listAcct:availableToPartners>
             <listAcct:currency>USA</listAcct:currency>
          </platformCore:record>
       </platformCore:recordList>
    </platformCore:searchResult>
 </searchResponse>
</soapenv:Body>
```

Transaction Search

Nearly all transaction record types use the TransactionSearch record for search. The *basic* element in this record references the TransactionSearchBasic record, which lists all available search filter fields available in a transaction search.

The TransactionSearch record also lists all search joins available in a transaction search. For details, see the SuiteTalk Schema Browser's TransactionSearch reference page.

Be aware that the actual search filter fields available will vary depending on the transaction type you are searching against. Not *all* of the search filter fields defined in TransactionSearchBasic will exist on every single transaction type. For example, the Check transaction type may include search filter fields not available on the Journal Entry type. However, search filters values for both record types will be defined in TransactionSearchBasic.

A TransactionSearchAdvanced request with specified columns returns every tranLine as a record. For example, if a sales order contains 1 item line, search results include up to 3 records, depending on the accounts that are touched. This behavior is the same behavior as transaction search in the UI. By contrast, TransactionSearchBasic, and TransactionSearchAdvanced without specified columns, return the whole transaction record. For example, if a search returns a sales order with 3 item lines, search results consist of one record, with ItemList containing 3 items. These operations return basically the same results as a get() operation.

Note: The Opportunity and the Time Bill records are the only transaction types that have their own search interfaces. For details, see the Schema Browser's OpportunitySearch and TimeBillSearch reference pages.

```
// the sales order
RecordRef salesOrderRef = new RecordRef();
salesOrderRef.setInternalId("873");
TransactionSearchBasic tranSearch = new TransactionSearchBasic();
// type is item fulfill
tranSearch.setType( new SearchEnumMultiSelectField( new String [] {RecordType.itemFulfillment.g
etValue()}, SearchEnumMultiSelectFieldOperator.anyOf));
// created from our sales order
tranSearch.setCreatedFrom( new SearchMultiSelectField( new RecordRef [] { salesOrderRef }, Sear
chMultiSelectFieldOperator.anyOf));
SearchResult result = c.search(tranSearch);
And the outgoing search soap looks like this:
<search xmlns="urn:messages_2017_1.platform.webservices.netsu ite.com">
<searchRecord xsi:type="ns1:TransactionSearchBasic" xmlns:ns1="urn:common_2017_1.platform.webse
rvices.netsuite.com">
<ns1:createdFrom operator="anyOf" xsi:type="ns2:SearchMultiSelectField" xmlns:ns2="urn:core_201
7_1.platform.webservices.netsuite.com">
<ns2:searchValue internalId="873" xsi:type="ns2:RecordRef"/>
</ns1:createdFrom>
<ns1:type operator="anyOf" xsi:type="ns3:SearchEnumMultiSelectField" xmlns:ns3="urn:core_2017_1
.platform.webservices.netsuite.com">
<ns3:searchValue xsi:type="xsd:string">itemFulfillment</ns3:searchValue>
</ns1:type>
</searchRecord>
</search>
```

Important: By default only a record's body fields are returned on a search. Therefore, you must set the **bodyFieldsOnly** element of the SearchPreferences type to **false** if you want to also return the information specified on a record's sublist. For general information on searching in web services, see the help topic search.

Sample Code

The following sample returns three records, including one that was modified within two minutes the search was made.

SOAP Request

```
 <soapenv:Body>
 <search xmlns="urn:messages_2017_1.platform.webservices.netsu ite.com">
 <searchRecord xsi:type="ns9:TransactionSearchBasic" xmlns:ns9="urn:common_2017_1.platform.webs
ervices.netsuite.com">
 <ns9:lastModifiedDate operator="onOrAfter" xsi:type="ns10:SearchDateField" xmlns:ns10="urn:cor
e_2017_1.platform.webservices.nets uite.com">
<ns10:predefinedSearchValue xsi:type="ns11:SearchDate" xmlns:ns11="urn:types.core_2017_1.platfo
rm.webservices.netsuite.com">today</ns10:predefinedSearchValue>
</ns9:lastModifiedDate>
 <ns9:type operator="anyOf" xsi:type="ns12:SearchEnumMultiSelectField" xmlns:ns12="urn:core_201
7_1.platform.webservices.netsuite.com">
<ns12:searchValue xsi:type="xsd:string">_salesOrder</ns12:searchValue>
</ns9:type>
</searchRecord>
</search>
</soapenv:Body>
```

Java

```
TransactionSearchBasic basic = new TransactionSearchBasic();
```

```
SearchEnumMultiSelectField soType = new SearchEnumMultiSelectField();
soType.setSearchValue(new String[1]);
soType.setSearchValue(0, TransactionType.__salesOrder);
soType.setOperator(SearchEnumMultiSelectFieldOperator.anyOf);
basic.setType(soType);

SearchDateField todayLastMod = new SearchDateField();
todayLastMod.setOperator(SearchDateFieldOperator.o nOrAfter);
todayLastMod.setPredefinedSearchValue(SearchDate.t oday);

basic.setLastModifiedDate(todayLastMod);
```

Country, State, and Language Enumerations

The following sections list all country and language enumerations defined in SuiteTalk, as well as explain how to set and validate state values in your web services requests:

- Country Enumerations
- Setting State Values in Web Services
- Language Enumerations

Important: Be aware that NetSuite uses country/state validation. This means that if the *Allow Free-Form States in Addresses* preference is set to FALSE, the state value is validated against the country. For example, you would not be able to set **CA** for **UK** since California is not a state that exists within the United Kingdom. For information on setting the *Allow Free-Form States in Addresses* preference, see Setting the Allow Free-Form States in Addresses Preference.

As a general rule, enumerations are camel-cased concatenations of the country name as defined in the NetSuite UI.

Country Enumerations

The following table lists countries, as defined in NetSuite dropdown lists, and the corresponding enumeration to be used when populating these fields in web services requests.

The country codes listed here are the ISO compliant country code string values returned when performing a search operation on entity records, whereas the schema enumeration is the corresponding value used during a request. These enumerations are also listed as common.xsd types in the SuiteTalk Schema Browser.

These enumerations can be used for all standard country fields in web services operations. These enumerations do not work in web services operations on custom fields that reference a List/Record of Country. In these cases, the internal ID of the country (country code) must be used. The internal ID can be obtained by calling the getSelectValue operation. This operation allows the creation of a mapping of country values with their internal IDs.

Note: The countryofmanufacture field for an item must be a country covered by FedEx or UPS.

Country	Schema Enumeration	Country Code
Afghanistan	_afghanistan	AF
Aland Islands	_alandIslands	AX
Albania	_albania	AL
Algeria	_algeria	DZ
American Samoa	_americanSamoa	AS
Andorra	_andorra	AD
Angola	_angola	AO
Anguilla	_anguilla	AI
Antarctica	_antarctica	AQ

Country	Schema Enumeration	Country Code
Antigua and Barbuda	_antiguaAndBarbuda	AG
Argentina	_argentina	AR
Armenia	_armenia	AM
Aruba	_aruba	AW
Australia	_australia	AU
Austria	_austria	AT
Azerbaijan	_azerbaijan	AZ
Bahamas	_bahamas	BS
Bahrain	_bahrain	BH
Bangladesh	_bangladesh	BD
Barbados	_barbados	BB
Belarus	_belarus	BY
Belgium	_belgium	BE
Belize	_belize	BZ
Benin	_benin	BJ
Bermuda	_bermuda	BM
Bhutan	_bhutan	BT
Bolivia	_bolivia	BO
Bonaire, Saint Eustatius, and Saba	_bonaireSaintEustatiusAndSaba	BQ
Bosnia and Herzegovina	_bosniaAndHerzegovina	BA
Botswana	_botswana	BW
Bouvet Island	_bouvetIsland	BV
Brazil	_brazil	BR
British Indian Ocean Territory	_britishIndianOceanTerritory	IO
Brunei Darussalam	_bruneiDarussalam	BN
Bulgaria	_bulgaria	BG
Burkina Faso	_burkinaFaso	BF
Burundi	_burundi	BI
Cambodia	_cambodia	KH
Cameroon	_cameroon	CM
Canada	_canada	CA
Canary Islands	_canaryIslands	IC
Cape Verde	_capeVerde	CV
Cayman Islands	_caymanIslands	KY
Central African Republic	_centralAfricanRepublic	CF
Ceuta and Melilla	_ceutaAndMelilla	EA

Country	Schema Enumeration	Country Code
Chad	_chad	TD
Chile	_chile	CL
China	_china	CN
Christmas Island	_christmasIsland	CX
Cocos (Keeling) Islands	_cocosKeelingIslands	CC
Colombia	_colombia	CO
Comoros	_comoros	KM
Congo, Democratic People's Republic	_congoDemocraticPeoplesRepublic	CD
Congo, Republic of	_congoRepublicOf	CG
Cook Islands	_cookIslands	CK
Costa Rica	_costaRica	CR
Cote d'Ivoire	_coteDIvoire	CI
Croatia/Hrvatska	_croatiaHrvatska	HR
Cuba	_cuba	CU
Curacao	_curacao	CW
Cyprus	_cyprus	CY
Czech Republic	_czechRepublic	CZ
Denmark	_denmark	DK
Djibouti	_djibouti	DJ
Dominica	_dominica	DM
Dominican Republic	_dominicanRepublic	DO
East Timor	_eastTimor	TP
Ecuador	_ecuador	EC
Egypt	_egypt	EG
El Salvador	_elSalvador	SV
Equatorial Guinea	_equatorialGuinea	GQ
Eritrea	_eritrea	ER
Estonia	_estonia	EE
Ethiopia	_ethiopia	ET
Falkland Islands	_falklandIslands	FK
Faroe Islands	_faroeIslands	FO
Fiji	_fiji	FJ
Finland	_finland	FI
France	_france	FR
French Guiana	_frenchGuiana	GF
French Polynesia	_frenchPolynesia	PF

Country	Schema Enumeration	Country Code
French Southern Territories	_frenchSouthernTerritories	TF
Gabon	_gabon	GA
Gambia	_gambia	GM
Georgia	_georgia	GE
Germany	_germany	DE
Ghana	_ghana	GH
Gibraltar	_gibraltar	GI
Greece	_greece	GR
Greenland	_greenland	GL
Grenada	_grenada	GD
Guadeloupe	_guadeloupe	GP
Guam	_guam	GU
Guatemala	_guatemala	GT
Guernsey	_guernsey	GG
Guinea	_guinea	GN
Guinea-Bissau	_guineaBissau	GW
Guyana	_guyana	GY
Haiti	_haiti	HT
Heard and McDonald Islands	_heardAndMcDonaldIslands	HM
Holy See (City Vatican State)	_holySeeCityVaticanState	VA
Honduras	_honduras	HN
Hong Kong	_hongKong	HK
Hungary	_hungary	HU
Iceland	_iceland	IS
India	_india	IN
Indonesia	_indonesia	ID
Iran (Islamic Republic of)	_iranIslamicRepublicOf	IR
Iraq	_iraq	IQ
Ireland	_ireland	IE
Isle of Man	_isleOfMan	IM
Israel	_israel	IL
Italy	_italy	IT
Jamaica	_jamaica	JM
Japan	_japan	JP
Jersey	_jersey	JE
Jordan	_jordan	JO

Country	Schema Enumeration	Country Code
Kazakhstan	_kazakhstan	KZ
Kenya	_kenya	KE
Kiribati	_kiribati	KI
Korea, Democratic People's Republic	_koreaDemocraticPeoplesRepublic	KP
Korea, Republic of	_koreaRepublicOf	KR
Kosovo	_kosovo	XK
Kuwait	_kuwait	KW
Kyrgyzstan	_kyrgyzstan	KG
Lao, People's Democratic Republic	_laoPeoplesDemocraticRepublic	LA
Latvia	_latvia	LV
Lebanon	_lebanon	LB
Lesotho	_lesotho	LS
Liberia	_liberia	LR
Libya	_libya	LY
Liechtenstein	_liechtenstein	LI
Lithuania	_lithuania	LT
Luxembourg	_luxembourg	LU
Macau	_macau	MO
Macedonia	_macedonia	MK
Madagascar	_madagascar	MG
Malawi	_malawi	MW
Malaysia	_malaysia	MY
Maldives	_maldives	MV
Mali	_mali	ML
Malta	_malta	MT
Marshall Islands	_marshallIslands	MH
Martinique	_martinique	MQ
Mauritania	_mauritania	MR
Mauritius	_mauritius	MU
Mayotte	_mayotte	YT
Mexico	_mexico	MX
Micronesia, Federal State of	_micronesiaFederalStateOf	FM
Moldova, Republic of	_moldovaRepublicOf	MD
Monaco	_monaco	MC
Mongolia	_mongolia	MN
Montenegro	_montenegro	ME

Country	Schema Enumeration	Country Code
Montserrat	_montserrat	MS
Morocco	_morocco	MA
Mozambique	_mozambique	MZ
Myanmar	_myanmar	MM
Namibia	_namibia	NA
Nauru	_nauru	NR
Nepal	_nepal	NP
Netherlands	_netherlands	NL
New Caledonia	_newCaledonia	NC
New Zealand	_newZealand	NZ
Nicaragua	_nicaragua	NI
Niger	_niger	NE
Nigeria	_nigeria	NG
Niue	_niue	NU
Norfolk Island	_norfolkIsland	NF
Northern Mariana Islands	_northernMarianaIslands	MP
Norway	_norway	NO
Oman	_oman	OM
Pakistan	_pakistan	PK
Palau	_palau	PW
Panama	_panama	PA
Papua New Guinea	_papuaNewGuinea	PG
Paraguay	_paraguay	PY
Peru	_peru	PE
Philippines	_philippines	PH
Pitcairn Island	_pitcairnIsland	PN
Poland	_poland	PL
Portugal	_portugal	PT
Puerto Rico	_puertoRico	PR
Qatar	_qatar	QA
Reunion Island	_reunionIsland	RE
Romania	_romania	RO
Russian Federation	_russianFederation	RU
Rwanda	_rwanda	RW
Saint Barthélemy	_saintBarthelemy	BL
Saint Helena	_saintHelena	SH

Country	Schema Enumeration	Country Code
Saint Kitts and Nevis	_saintKittsAndNevis	KN
Saint Lucia	_saintLucia	LC
Saint Martin	_saintMartin	MF
Saint Vincent and the Grenadines	_saintVincentAndTheGrenadines	VC
Samoa	_samoa	WS
San Marino	_sanMarino	SM
Sao Tome and Principe	_saoTomeAndPrincipe	ST
Saudi Arabia	_saudiArabia	SA
Senegal	_senegal	SN
Serbia	_serbia	RS
Seychelles	_seychelles	SC
Sierra Leone	_sierraLeone	SL
Singapore	_singapore	SG
Sint Maarten	_sintMaarten	SX
Slovak Republic	_slovakRepublic	SK
Slovenia	_slovenia	SI
Solomon Islands	_solomonIslands	SB
Somalia	_somalia	SO
South Africa	_southAfrica	ZA
South Georgia	_southGeorgia	GS
South Sudan	_southSudan	SS
Spain	_spain	ES
Sri Lanka	_sriLanka	LK
State of Palestine	_stateOfPalestine	PS
St. Pierre and Miquelon	_stPierreAndMiquelon	PM
Sudan	_sudan	SD
Suriname	_suriname	SR
Svalbard and Jan Mayen Islands	_svalbardAndJanMayenIslands	SJ
Swaziland	_swaziland	SZ
Sweden	_sweden	SE
Switzerland	_switzerland	CH
Syrian Arab Republic	_syrianArabRepublic	SY
Taiwan	_taiwan	TW
Tajikistan	_tajikistan	TJ
Tanzania	_tanzania	TZ
Thailand	_thailand	TH

Country	Schema Enumeration	Country Code
Togo	_togo	TG
Tokelau	_tokelau	TK
Tonga	_tonga	TO
Trinidad and Tobago	_trinidadAndTobago	TT
Tunisia	_tunisia	TN
Turkey	_turkey	TR
Turkmenistan	_turkmenistan	TM
Turks and Caicos Islands	_turksAndCaicosIslands	TC
Tuvalu	_tuvalu	TV
Uganda	_uganda	UG
Ukraine	_ukraine	UA
United Arab Emirates	_unitedArabEmirates	AE
United Kingdom	_unitedKingdom	GB
United States	_unitedStates	US
Uruguay	_uruguay	UY
US Minor Outlying Islands	_uSMinorOutlyingIslands	UM
Uzbekistan	_uzbekistan	UZ
Vanuatu	_vanuatu	VU
Venezuela	_venezuela	VE
Vietnam	_vietnam	VN
Virgin Islands, British	_virginIslandsBritish	VG
Virgin Islands, USA	_virginIslandsUSA	VI
Wallis and Futuna Islands	_wallisAndFutunaIslands	WF
Western Sahara	_westernSahara	EH
Yemen	_yemen	YE
Zambia	_zambia	ZM
Zimbabwe	_zimbabwe	ZW

Setting State Values in Web Services

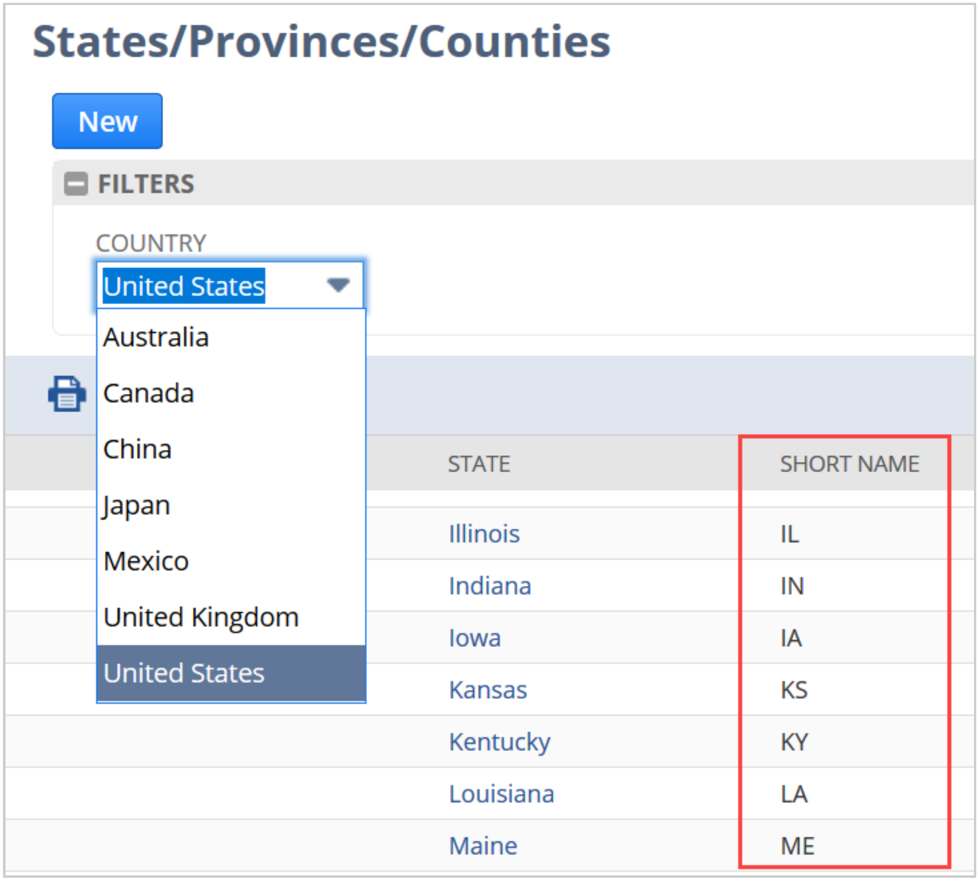
The WSDL does not include a State class or state enumeration values (previously defined in commonTypes.xsd in endpoints older than 2008.2). State fields now take any of the string values defined in the **Short Name** column on the State/Provinces/Countries page in the NetSuite user interface (see figure).

- Navigate to this page by going to Setup > Company > States/Provinces/Counties.

Note that some short name values may contain characters that are not translated in some languages. You can retrieve a list of translated short names through web services. However, in web services, you cannot use the search operation to retrieve state values. You must use the **getAll** operation to retrieve all state values in the system. This 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.

Be aware that state values can be added, updated, and deleted through the NetSuite user interface and through web services. For information about working with states, provinces, and counties in the user interface, see the help topic Setting Up States, Provinces, and Counties.

State and province names in web services must match the short name values listed at Setup > Company > States/Provinces/Counties.



STATE	SHORT NAME
Illinois	IL
Indiana	IN
Iowa	IA
Kansas	KS
Kentucky	KY
Louisiana	LA
Maine	ME

The Java and SOAP samples below show how to create a new customer and define all address properties on the customer Address (addressbook) sublist. Note that the state value is defined as the string " CA " — as it appears on the States/Provinces/Counties page in the UI.

Java

```
public void add_customer_with_address() throws Exception {
Customer c = new Customer();
c.setCompanyName("my company");
CustomerAddressbook cab = new CustomerAddressbook();
cab.setAddr1("123 Main St");
cab.setCity("San Mateo");
cab.setState("CA");
cab.setZip("94403");
cab.setCountry(Country._unitedStates);
c.setAddressbookList(new CustomerAddressbookList(new
   CustomerAddressbook[]{cab}, false));
sessMgr.getPort().add(c);
}
```

SOAP

```
<add xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
            <record externalId="Strawberry10150306" xsi:type="ns1:Customer"
      xmlns:ns1="urn:relationships_2017_1.lists.webservices.netsuite.com">
               <ns1:companyName xsi:type="xsd:string">my company</ns1:companyName>
               <ns1:addressbookList replaceAll="false" xsi:type="ns1:CustomerAddressbookList">
                  <ns1:addressbook xsi:type="ns1:CustomerAddressbook">
                     <ns1:addr1 xsi:type="xsd:string">123 Main St</ns1:addr1>
                     <ns1:city xsi:type="xsd:string">San Mateo</ns1:city>
                     <ns1:zip xsi:type="xsd:string">94403</ns1:zip>
                     <ns1:country xsi:type="ns2:Country"
         xmlns:ns2="urn:types.common_2017_1.platform.
            webservices.netsuite.com">_unitedStates
         <ns1:country>
                     <ns1:state xsi:type="xsd:string">CA</ns1:state>
                  </ns1:addressbook>
               </ns1:addressbookList>
            </record>
         </add>
```

Important: State enumeration changes made in the 2008.2 endpoint have no impact on WSDL versions 2008.1 and lower. However, users *will* have to change their code when they upgrade to the 2009.1 endpoint.

For example, when upgrading to the 2009.1 endpoint, existing code that reads something similar to:

```
address.state = State._california;
```

should be changed to:

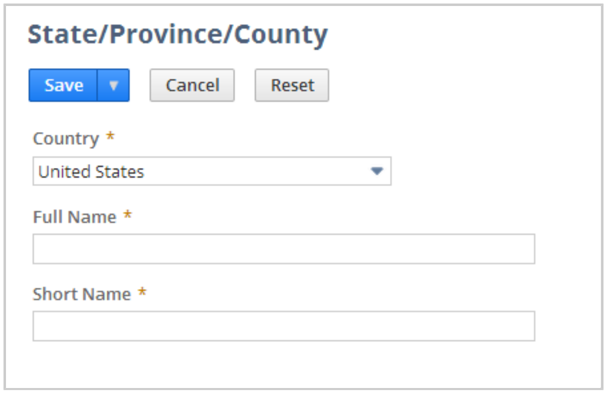
```
address.state("CA");
```

Creating Custom State Values

For countries that do not have country/state values built in to NetSuite (for example, Portugal or Argentina), you can create a list of custom state values and assign these values to a specific country. Doing so enables you to enforce country/state validation when the *Allow Free-Form States in Addresses* is set to FALSE.

Note: See Setting the Allow Free-Form States in Addresses Preference for more information on setting this preference and enabling or disabling country/state validation.

Create custom state values by going to Setup > Company > States/Provinces/Counties > New. In the Country dropdown list, select your country. In the Full Name and Short Name fields, specify your state values. Note that you must use the **Short Name** value in your web services requests.



Country *

United States

Full Name *

Short Name *

Note: You can also click the New button at the top of the State/Province/Countries list page to create custom states.

Setting the Allow Free-Form States in Addresses Preference

Selecting the **Allow Free-Form States in Addresses** preference controls whether country/state validation is performed during a web services request. In the user interface, you navigate to this preference by going to Setup > Company > General Preferences.

To enable country/state validation:

1. Make sure the **Allow Free-Form States in Addresses** box is not checked. When the box is not checked, the system validates the state short name (for example, CA or GA) against the country.
2. In your web services request, you must then submit the abbreviation that exists in the Setup > Company > States/Provinces/Counties list, in the **Short Name** column.
3. If the country/state values do not exist in the user interface, extend this list by going to Setup > Company > State/Provinces/Countries > New to define custom states and associate them with a country. (See Creating Custom State Values for more information on creating custom state values.)

To disable country/state validation:

1. Make sure the **Allow Free-Form States in Addresses** box is checked
2. In your web services request, send any state value in the request. NetSuite stores the value you send as a string.

Language Enumerations

The following table lists languages that can be set through web services and returned as the user's preferred language.

UI language names listed in this table match those displayed as user and company language preferences in the NetSuite user interface. Schema enumerations are the corresponding values used during a request. These enumerations are also listed as common.xsd types in the SuiteTalk Schema Browser.

For system-supported languages, NetSuite provides translation strings for the user interface and for printed transaction forms. The Multi-Language feature also supports the use of several additional languages for website translations, but these languages must be defined in company preferences and translation strings must be provided. See the help topic Configuring Multiple Languages.

UI Language Name	Schema Enumeration	System-Supported?
Afrikaans	_afrikaans	No
Albanian	_albanian	No
Arabic	_arabic	No
Armenian	_armenian	No
Bengali	_bengali	No
Bosnian	_bosnian	No
Bulgarian	_bulgarian	No
(not available in UI)	_catalan	No
Chinese (Simplified)	_chineseSimplified	Yes
Chinese (Traditional)	_chineseTraditional	Yes
Croatian	_croatian	No
Czech	_czech	Yes
Danish	_danish	Yes
Dutch	_dutch	Yes
English (AU)	_englishAu	Yes
English (CA)	_englishCa	Yes
English (International)	_englishInternational	Yes
English (UK)	_englishUK	Yes
English (US)	_englishUS	Yes
Estonian	_estonian	No
Filipino	_filipino	No
Finnish	_finnish	No
French (Canada)	_frenchCanada	Yes
French	_frenchFrance	Yes
German	_german	Yes
Greek	_greek	No
Gujarati	_gujarati	No
(not available in UI)	_haitian	No
Hebrew	_hebrew	No
Hindi	_hindi	No
Hungarian	_hungarian	No
Icelandic	_icelandic	No
Indonesian	_indonesian	No

UI Language Name	Schema Enumeration	System-Supported?
Italian	_italian	Yes
Japanese	_japanese	Yes
Kannada	_kannada	No
Korean	_korean	Yes
Spanish (Latin America)	_latinAmericanSpanish	Yes
Latvian	_latvian	No
Lithuanian	_lithuanian	No
Luxembourgish	_luxembourgish	No
Malay	_malay	No
Marathi	_marathi	No
Norwegian	_norwegian	No
Persian (Iran)	_persianIran	No
Polish	_polish	No
Portuguese (Brazil)	_portugueseBrazil	Yes
Portuguese (Portugal)	_portuguesePortugal	No
Punjabi	_punjabi	No
Romanian	_romanian	No
Russian	_russian	Yes
Serbian (Cyrillic)	_serbianCyrillic	No
Serbian (Latin)	_serbianLatin	No
Slovak	_slovak	No
Slovenian	_slovenian	No
Spanish	_spanish	Yes
Swedish	_swedish	Yes
Tamil	_tamil	No
Telugu	_telugu	No
Thai	_thai	Yes
Turkish	_turkish	Yes
Ukrainian	_ukrainian	No
Vietnamese	_vietnamese	No