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

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

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

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

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

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

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

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

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

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

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

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your pre-General Availability trial agreement only. It is not a commitment to deliver any material, code, or functionality, and
should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Master Agreement, Oracle License and Services Agreement, Oracle PartnerNetwork Agreement, Oracle distribution agreement, or other license agreement which has been executed by you and Oracle and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

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

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

Sample Code

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

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.
Send Us Your Feedback

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

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

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

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

To report software issues, contact NetSuite Customer Support.
## Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuiteTalk SOAP Web Services Platform Overview</td>
<td>1</td>
</tr>
<tr>
<td>Support Has Ended for SOAP Web Services 2009.2 and Earlier Endpoints</td>
<td>2</td>
</tr>
<tr>
<td>SOAP Web Services Overview</td>
<td>3</td>
</tr>
<tr>
<td>NetSuite WSDL and XSD Structure</td>
<td>3</td>
</tr>
<tr>
<td>NetSuite WSDL Versioning</td>
<td>11</td>
</tr>
<tr>
<td>NetSuite Versioning and WSDL Versioning Overview</td>
<td>11</td>
</tr>
<tr>
<td>Upgrading WSDL Versions</td>
<td>11</td>
</tr>
<tr>
<td>Support for Existing WSDL Versions</td>
<td>13</td>
</tr>
<tr>
<td>SOAP Web Services Performance Optimization</td>
<td>14</td>
</tr>
<tr>
<td>SOAP Web Services Governance Overview</td>
<td>15</td>
</tr>
<tr>
<td>Web Services and RESTlet Concurrency Governance</td>
<td>16</td>
</tr>
<tr>
<td>Concurrency Governance Limits Based on Service Tiers and SuiteCloud Plus Licenses</td>
<td>17</td>
</tr>
<tr>
<td>Concurrency Limit per Integration</td>
<td>18</td>
</tr>
<tr>
<td>Concurrency Governance for Internal Applications</td>
<td>19</td>
</tr>
<tr>
<td>Effects of Authentication Method on Concurrency Governance</td>
<td>20</td>
</tr>
<tr>
<td>Account Concurrency Monitoring Resources</td>
<td>20</td>
</tr>
<tr>
<td>Testing Your Concurrency Usage</td>
<td>21</td>
</tr>
<tr>
<td>Errors Related to Concurrency Violations</td>
<td>21</td>
</tr>
<tr>
<td>Recommended Actions</td>
<td>21</td>
</tr>
<tr>
<td>Retrying Failed Web Services Requests</td>
<td>22</td>
</tr>
<tr>
<td>Sample Scenario</td>
<td>22</td>
</tr>
<tr>
<td>Frequently Asked Questions</td>
<td>23</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>24</td>
</tr>
<tr>
<td>SOAP Web Services Setup</td>
<td>25</td>
</tr>
<tr>
<td>SOAP Web Services Quick Start</td>
<td>25</td>
</tr>
<tr>
<td>Enabling the SOAP Web Services Feature</td>
<td>25</td>
</tr>
<tr>
<td>Enabling the Show Internal IDs Preference</td>
<td>26</td>
</tr>
<tr>
<td>Building an Application with Microsoft .NET</td>
<td>26</td>
</tr>
<tr>
<td>Building an Application with Java using Apache Axis</td>
<td>28</td>
</tr>
<tr>
<td>Downloading Sample Applications</td>
<td>30</td>
</tr>
<tr>
<td>Capturing SOAP</td>
<td>30</td>
</tr>
<tr>
<td>SOAP Web Services Development Considerations</td>
<td>31</td>
</tr>
<tr>
<td>Development Considerations Overview</td>
<td>31</td>
</tr>
<tr>
<td>NetSuite Features in SOAP Web Services</td>
<td>32</td>
</tr>
<tr>
<td>SOAP Web Services Reliability Considerations</td>
<td>32</td>
</tr>
<tr>
<td>Effects of Account Configuration in SOAP Web Services</td>
<td>34</td>
</tr>
<tr>
<td>Effects of Accounts' Host Data Center on URLs</td>
<td>34</td>
</tr>
<tr>
<td>Enumerations and Special Characters</td>
<td>38</td>
</tr>
<tr>
<td>Image References in SOAP Web Services</td>
<td>38</td>
</tr>
<tr>
<td>SOAP Web Services Preferences</td>
<td>39</td>
</tr>
<tr>
<td>Company-Wide Preferences</td>
<td>39</td>
</tr>
<tr>
<td>One-Time Preferences</td>
<td>41</td>
</tr>
<tr>
<td>Request-Level Preferences</td>
<td>41</td>
</tr>
<tr>
<td>Sample Code</td>
<td>45</td>
</tr>
<tr>
<td>Reseting Default Behavior</td>
<td>46</td>
</tr>
<tr>
<td>Setting the Internal ID Preference</td>
<td>48</td>
</tr>
<tr>
<td>Caching Behavior in SOAP Web Services</td>
<td>48</td>
</tr>
<tr>
<td>Integration Management</td>
<td>49</td>
</tr>
<tr>
<td>Integration Record Overview</td>
<td>49</td>
</tr>
<tr>
<td>Adding an Integration Record</td>
<td>52</td>
</tr>
<tr>
<td>Creating an Integration Record</td>
<td>52</td>
</tr>
<tr>
<td>Auto-Installation of Integration Records</td>
<td>54</td>
</tr>
<tr>
<td>Bundle Installation of Integration Records</td>
<td>55</td>
</tr>
<tr>
<td>Topic</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Advanced Searches in SOAP Web Services</td>
<td>269</td>
</tr>
<tr>
<td>Joining Through Custom Fields</td>
<td>279</td>
</tr>
<tr>
<td>Setting Valid Search Values</td>
<td>283</td>
</tr>
<tr>
<td>Setting the anyof, mine, or myteam Filtering Values</td>
<td>284</td>
</tr>
<tr>
<td>Searching by lastModifiedDate</td>
<td>285</td>
</tr>
<tr>
<td>Understanding Sorting in Advanced Search</td>
<td>286</td>
</tr>
<tr>
<td>Search-Related Sample Code</td>
<td>288</td>
</tr>
<tr>
<td>Searching for a Multi-select Custom Field</td>
<td>292</td>
</tr>
<tr>
<td>Search Issues and Best Practices for SOAP Web Services and SuiteScript</td>
<td>303</td>
</tr>
<tr>
<td>searchMore</td>
<td>304</td>
</tr>
<tr>
<td>searchMoreWithId</td>
<td>309</td>
</tr>
<tr>
<td>searchNext</td>
<td>314</td>
</tr>
<tr>
<td>ssoLogin</td>
<td>316</td>
</tr>
<tr>
<td>update</td>
<td>319</td>
</tr>
<tr>
<td>updateList</td>
<td>323</td>
</tr>
<tr>
<td>updateInviteeStatus</td>
<td>327</td>
</tr>
<tr>
<td>updateInviteeStatusList</td>
<td>330</td>
</tr>
<tr>
<td>upsert</td>
<td>332</td>
</tr>
<tr>
<td>upsertList</td>
<td>335</td>
</tr>
<tr>
<td>SOAP Web Services PHP Toolkit</td>
<td>341</td>
</tr>
<tr>
<td>PHP Toolkit Overview</td>
<td>341</td>
</tr>
<tr>
<td>Downloading the PHP Toolkit</td>
<td>343</td>
</tr>
<tr>
<td>Configuring an Environment for the PHP Toolkit</td>
<td>343</td>
</tr>
<tr>
<td>Creating a SOAP Web Services PHP Project</td>
<td>345</td>
</tr>
<tr>
<td>Creating and Submitting Records Using the PHP Toolkit</td>
<td>346</td>
</tr>
<tr>
<td>Logging SOAP Requests and Responses Using the PHP Toolkit</td>
<td>348</td>
</tr>
<tr>
<td>Troubleshooting PHP and SOAP Web Services</td>
<td>348</td>
</tr>
<tr>
<td>SOAP Web Services Error Handling and Error Codes</td>
<td>350</td>
</tr>
<tr>
<td>SOAP Web Services Warnings, Errors, and Faults</td>
<td>350</td>
</tr>
<tr>
<td>SOAP Faults for Each Operation</td>
<td>353</td>
</tr>
<tr>
<td>SOAP Fault Status Codes</td>
<td>356</td>
</tr>
<tr>
<td>Error Status Codes</td>
<td>357</td>
</tr>
<tr>
<td>Warning Status Codes</td>
<td>428</td>
</tr>
<tr>
<td>Frequently Asked Questions</td>
<td>429</td>
</tr>
<tr>
<td>SOAP Web Services FAQ</td>
<td>429</td>
</tr>
</tbody>
</table>
SuiteTalk SOAP Web Services Platform Overview

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

- **SOAP encoded web services** – the Platform uses SOAP-based web services with document style, or Doc style, encoding.
  
  Doc style encoding consists of message-oriented exchanges where an XML schema specified within the SOAP message defines the structure of any messages sent between two applications. RPC exchanges are NOT supported.

- **HTTPS Transport** – currently the only transport supported is HTTPS as defined in the WSDL document.

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

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

**Important:** Although the following material pertains to the 2019.2 WSDL, NetSuite continues to support endpoints released prior to 2019.2. For documentation about older endpoints that are supported, see the help topic [SOAP Web Services Archives](#).

In This Guide

This manual contains the following sections:

- **SOAP Web Services Setup** — provides steps for getting your SOAP web services development environment set up to create your first service. It also describes how to modify or set SOAP web services preferences. This section also describes general concepts that pertain to SOAP web services development.

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

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

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

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

- **SOAP Web Services Processing** — describes how to process requests synchronously versus asynchronously. Also provided are steps for monitoring your SOAP web services requests.
Support Has Ended for SOAP Web Services 2009.2 and Earlier Endpoints

NetSuite no longer supports SOAP web services 2009.2 or earlier endpoints. NetSuite strongly encourages customers and partners using 2009.2 and earlier versions of the SOAP web services WSDL to begin upgrading to later endpoints.

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

Determining Endpoints that are Currently in Use

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

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

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

Resources for Endpoint Upgrades

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

- For the schema of the latest endpoint, see: https://webservices.netsuite.com/wsdlv2019_2_0/netsuite.wsdl
- For details about the records supported in the latest endpoint, see the help topic SuiteTalk SOAP Records Overview.
Support Has Ended for SOAP Web Services 2009.2 and Earlier Endpoints  

- For general information about using SOAP web services, see SuiteTalk SOAP Web Services Platform Overview.
- For upgrade tips, see Upgrading WSDL Versions.
- If you have issues during your transition process, please contact NetSuite Customer Support.

Note About Record Processing Behavior in 2010.1 and Later Endpoints

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

SOAP Web Services Overview

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

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

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

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

NetSuite WSDL and XSD Structure

SOAP web services use single WSDL file that describes all supported operations and messages. You can access that file at the following link:

https://webservices.netsuite.com/wsdl/v2019_2_0/netsuite.wsdl

In this link, v2019_2_0 reflects the WSDL version.

NetSuite defines WSDL versioning, the location of schemas, namespaces, and the endpoint as follows:
WSDL: https://webservices.netsuite.com/wsd1/v2017_1_0/netsuite.wsdl

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

**Note:** For more information on WSDL versioning, upgrading, and testing, see NetSuite WSDL Versioning. For information on WSDLs that pre-date 2019.2, see the help topic SOAP Web Services Archives.

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

**Important:** NetSuite hosts customer accounts in multiple data centers. For that reason, the correct URL for SOAP web services access varies depending on the data center hosting the account.

For details about dynamic discovery of URLs, see Effects of Accounts' Host Data Center on URLs and Dynamic Discovery of URLs for SOAP Web Services and RESTlet Clients.

Your integration must incorporate logic that dynamically determines the correct URL. With the 2012.2 and later endpoints, you should use the getDataCenterUrls operation to dynamically discover the correct URL. With older endpoints, you should use the REST roles service. For details, see The REST roles Service. You can also use the DataCenterUrls REST service to dynamically discover the correct domain URLs.

### Messaging XSD Files

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

<table>
<thead>
<tr>
<th>URL</th>
<th>Schema Alias</th>
<th>Notes</th>
</tr>
</thead>
</table>
| https://webservices.netsuite.com/xsd/platform/v2019_2_0/common.xsd | platformCommon | This file also includes descriptions for the following subrecords:  
  - Address  
  - InventoryDetail  
  - LandedCost |
| https://webservices.netsuite.com/xsd/platform/v2019_2_0/core.xsd | platformCore | Among other elements, this file includes descriptions for the following complex types, which are used for authentication:  
  - Passport  
  - SsoPassport  
  - TokenPassport |
| https://webservices.netsuite.com/xsd/platform/v2019_2_0/faults.xsd | platformFaults |  |
| https://webservices.netsuite.com/xsd/platform/v2019_2_0/messages.xsd | platformMsgs |  |
Record Type Definitions

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

<table>
<thead>
<tr>
<th>URL</th>
<th>Schema Alias</th>
<th>Record Types</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/activities/v2019_2_0/scheduling.xsd">https://webservices.netsuite.com/xsd/activities/v2019_2_0/scheduling.xsd</a></td>
<td>actSched</td>
<td>ContactCategory, CustomerCategory, SalesRole, PriceLevel, WinLossReason, Term, NoteType, PaymentMethod, CalendarEvent, CalendarEventSearch, CalendarEventSearchAdvanced, Task, TaskSearch, TaskSearchAdvanced, PhoneCall, PhoneCallSearch, PhoneCallSearchAdvanced, ProjectTask, Resource Allocation</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/transactions/v2019_2_0/demandPlanning.xsd">https://webservices.netsuite.com/xsd/transactions/v2019_2_0/demandPlanning.xsd</a></td>
<td>demandPlanning</td>
<td>ItemDemandPlan, ItemSupplyPlan</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/documents/v2019_2_0/fileCabinet.xsd">https://webservices.netsuite.com/xsd/documents/v2019_2_0/fileCabinet.xsd</a></td>
<td>docfileCab</td>
<td>File, FileSearch, FileSearchAdvanced, Folder, FolderSearch, FolderSearchAdvanced</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/general/v2019_2_0/communication.xsd">https://webservices.netsuite.com/xsd/general/v2019_2_0/communication.xsd</a></td>
<td>generalComm</td>
<td>Note, NoteSearch, NoteSearchAdvanced, Message, MessageSearch, MessageSearchAdvanced</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/lists/v2019_2_0/accounting.xsd">https://webservices.netsuite.com/xsd/lists/v2019_2_0/accounting.xsd</a></td>
<td>listAcct</td>
<td>Billing Schedule, Bill of Materials (BOM), BOM Revision, ContactCategory, CustomerCategory, SalesRole, PriceLevel, WinLossReason, Term, NoteType, PaymentMethod, LeadSource, Assembly Item (BOM Item), DescriptionItem, DiscountItem, Download Item, GeneralToken, GiftCertificateItem, InventoryItem</td>
</tr>
<tr>
<td>URL</td>
<td>Schema Alias</td>
<td>Record Types</td>
</tr>
<tr>
<td>-----</td>
<td>--------------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
| https://webservices.netsuite.com/xsd/lists/v2019_2_0/accounting.xsd | listAcct | ItemSearch
ItemSearchAdvanced
Kit/Package Item
Lot Numbered Assembly Item
Lot Numbered Inventory Item
MerchandiseHierarchyNode
MarkupItem
NonInventoryPurchaseItem
NonInventoryResaleItem
NonInventorySaleItem
OtherChargePurchaseItem
OtherChargeResaleItem
OtherChargeSaleItem
PaymentCard
PaymentCardToken
PaymentItem
SerializedAssemblyItem
Serialized Inventory Item
ServicePurchaseItem
ServiceResaleItem
ServiceSaleItem
SubtotalItem
Currency
ExpenseCategory
Account
AccountSearch
AccountSearchAdvanced
AccountingPeriod
Bin
BinSearch
BinSearchAdvanced
Classification
ClassificationSearch
ClassificationSearchAdvanced
Department
DepartmentSearch
DepartmentSearchAdvanced
Expense Category
Gift Certificate
FairValuePrice
GiftCertificateSearch
GiftCertificateSearchAdvanced
GlobalAccountMapping
Inventory Number
ItemAccountMapping
Location
LocationSearch
LocationSearchAdvanced
Nexus
Revenue Recognition Schedule
Revenue Recognition Template
Sales Tax Item
Sales Tax ItemSearchAdvanced
ContactRole
Bin
SalesTaxItem
TaxAcct
TaxGroup
TaxType
Subsidiary
SubsidiarySearch
SubsidiarySearchAdvanced
UnitsType
PartnerCategory

<table>
<thead>
<tr>
<th>URL</th>
<th>Schema Alias</th>
<th>Record Types</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/lists/v2019_2_0/employees.xsd">https://webservices.netsuite.com/xsd/lists/v2019_2_0/employees.xsd</a></td>
<td>listEmp</td>
<td>Employee, EmployeeSearch, EmployeeSearchAdvanced, HCM Job</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/lists/v2019_2_0/marketing.xsd">https://webservices.netsuite.com/xsd/lists/v2019_2_0/marketing.xsd</a></td>
<td>listMkt</td>
<td>Campaign, CampaignSearch, CampaignSearchAdvanced, CampaignCategory, CampaignAudience, CampaignFamily, CampaignSearchEngine, CampaignChannel, CampaignOffer, CampaignResponse, CampaignVertical, CampaignSubscription, PromotionCode, PromotionCodeSearch, PromotionCodeSearchAdvanced</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/lists/v2019_2_0/supplyChain.xsd">https://webservices.netsuite.com/xsd/lists/v2019_2_0/supplyChain.xsd</a></td>
<td>listScm</td>
<td>ManufacturingCostTemplate, ManufacturingOperationTask, ManufacturingRouting</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/lists/v2019_2_0/website.xsd">https://webservices.netsuite.com/xsd/lists/v2019_2_0/website.xsd</a></td>
<td>listSite</td>
<td>SiteCategory, SiteCategorySearch, SiteCategorySearchAdvanced</td>
</tr>
<tr>
<td>URL</td>
<td>Schema Alias</td>
<td>Record Types</td>
</tr>
<tr>
<td>--------------------------------------------------------------------</td>
<td>------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/setup/v2019_2_0/customization.xsd">https://webservices.netsuite.com/xsd/setup/v2019_2_0/customization.xsd</a></td>
<td>setupCustom</td>
<td>CustomRecord, CustomRecordSearch, CustomRecordSearchAdvanced, CustomList, CustomRecordType, EntityCustomField, CrmCustomField, OtherCustomField, ItemCustomField, TransactionBodyCustomField, TransactionColumnCustomField, ItemOptionCustomField, CustomRecordCustomField, CustomTransaction, CustomSegment</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/transactions/v2019_2_0/bank.xsd">https://webservices.netsuite.com/xsd/transactions/v2019_2_0/bank.xsd</a></td>
<td>tranBank</td>
<td>Check, Deposit</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/transactions/v2019_2_0/customers.xsd">https://webservices.netsuite.com/xsd/transactions/v2019_2_0/customers.xsd</a></td>
<td>tranCust</td>
<td>CashRefund, Charge, CustomerPayment, ReturnAuthorization, CreditMemo, CustomerRefund, CustomerDeposit, DepositApplication</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/transactions/v2019_2_0/financial.xsd">https://webservices.netsuite.com/xsd/transactions/v2019_2_0/financial.xsd</a></td>
<td>tranFin</td>
<td>Budget, BudgetSearch, BudgetSearchAdvanced</td>
</tr>
</tbody>
</table>
NetSuite WSDL and XSD Structure

<table>
<thead>
<tr>
<th>URL</th>
<th>Schema Alias</th>
<th>Record Types</th>
</tr>
</thead>
</table>
| https://webservices.netsuite.com/xsd/transactions/v2019_2_0/purchases.xsd | tranPurch | VendorBill  
PurchaseOrder  
ItemReceipt  
VendorPayment  
VendorCredit  
Purchas/Requisition  
InboundShipment |
| https://webservices.netsuite.com/xsd/transactions/v2019_2_0/sales.xsd | tranSales | AccountingTransactionSearch  
AccountingTransactionSearchAdvanced  
Opportunity  
OpportunitySearch  
OpportunitySearchAdvanced  
SalesOrder  
TransactionSearch  
TransactionSearchAdvanced  
ItemFulfillment  
Invoice  
CashSale  
Estimate |

System Constants XSD Files

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

<table>
<thead>
<tr>
<th>URL</th>
<th>Schema Alias</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/activities/v2019_2_0/schedulingTypes.xsd">https://webservices.netsuite.com/xsd/activities/v2019_2_0/schedulingTypes.xsd</a></td>
<td>actSchedTyp</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/transactions/v2019_2_0/demandPlanning.xsd">https://webservices.netsuite.com/xsd/transactions/v2019_2_0/demandPlanning.xsd</a></td>
<td>demandPlanningTyp</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/documents/v2019_2_0/fileCabinetTypes.xsd">https://webservices.netsuite.com/xsd/documents/v2019_2_0/fileCabinetTypes.xsd</a></td>
<td>docFileCabTyp</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/general/v2019_2_0/communicationTypes.xsd">https://webservices.netsuite.com/xsd/general/v2019_2_0/communicationTypes.xsd</a></td>
<td>generalCommTyp</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/transactions/v2019_2_0/inventoryTypes.xsd">https://webservices.netsuite.com/xsd/transactions/v2019_2_0/inventoryTypes.xsd</a></td>
<td>invtTyp</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/lists/v2019_2_0/accountingTypes.xsd">https://webservices.netsuite.com/xsd/lists/v2019_2_0/accountingTypes.xsd</a></td>
<td>listAcctTyp</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/lists/v2019_2_0/employeeTypes.xsd">https://webservices.netsuite.com/xsd/lists/v2019_2_0/employeeTypes.xsd</a></td>
<td>listEmpTyp</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/lists/v2019_2_0/marketingTypes.xsd">https://webservices.netsuite.com/xsd/lists/v2019_2_0/marketingTypes.xsd</a></td>
<td>listMktTyp</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/lists/v2019_2_0/relationshipTypes.xsd">https://webservices.netsuite.com/xsd/lists/v2019_2_0/relationshipTypes.xsd</a></td>
<td>listRelTyp</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/lists/v2019_2_0/supplyChainTypes.xsd">https://webservices.netsuite.com/xsd/lists/v2019_2_0/supplyChainTypes.xsd</a></td>
<td>listScmTyp</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/lists/v2019_2_0/supportTypes.xsd">https://webservices.netsuite.com/xsd/lists/v2019_2_0/supportTypes.xsd</a></td>
<td>listSupportTyp</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/platform/v2019_2_0/coreTypes.xsd">https://webservices.netsuite.com/xsd/platform/v2019_2_0/coreTypes.xsd</a></td>
<td>platformCoreTyp</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/platform/v2019_2_0/faultTypes.xsd">https://webservices.netsuite.com/xsd/platform/v2019_2_0/faultTypes.xsd</a></td>
<td>platformFaultsTyp</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/setup/v2019_2_0/customizationTypes.xsd">https://webservices.netsuite.com/xsd/setup/v2019_2_0/customizationTypes.xsd</a></td>
<td>setupCustomTyp</td>
</tr>
<tr>
<td><a href="https://webservices.netsuite.com/xsd/transactions/v2019_2_0/saleTypes.xsd">https://webservices.netsuite.com/xsd/transactions/v2019_2_0/saleTypes.xsd</a></td>
<td>tranSalesTyp</td>
</tr>
</tbody>
</table>

Example

For example, the addRequest message type has three levels of referencing.
In the WSDL file, the `addRequest` message is defined as:

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

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

https://webservices.netsuite.com/xsd/platform/v2019_2_0/messages.xsd

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

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

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

https://webservices.netsuite.com/xsd/platform/v2019_2_0/core.xsd

The abstract type Record is defined as:

```xml
<complexType name="Record" abstract="true">
  <sequence>
    <element name="nullFieldList" type="platformCore:NullField" minOccurs="0" maxOccurs="1" />
  </sequence>
</complexType>
```
NetSuite WSDL and XSD Structure

**Note:** In SOAP web services, Record is the base for all other NetSuite record types.

### NetSuite WSDL Versioning

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

- NetSuite Versioning and WSDL Versioning Overview
- Upgrading WSDL Versions
- Support for Existing WSDL Versions
- SOAP Web Services Archives

#### NetSuite Versioning and WSDL Versioning Overview

The naming convention for NetSuite versions and WSDL versions are the same. For example, for 2019.2, there is an accompanying 2019_2 version of the WSDL, as shown in the following WSDL URL:

`https://webservices.netsuite.com/wsdl/v2019_2_0/netsuite.wsdl`

NetSuite WSDL versioning denotes:

- the major version: `v2019_2`
- the minor or patch version: `_0`

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

Be aware that when your NetSuite account is upgraded, you do not have to upgrade your WSDL to the supporting NetSuite version. For example, you may still use 2019.1 of the WSDL in conjunction with a NetSuite account that has been upgraded to 2019.2. For more information, see Upgrading WSDL Versions.

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

#### Upgrading WSDL Versions

NetSuite customers typically upgrade their WSDL when a newer version includes functionality that enables them to meet a particular business need. They will also upgrade their WSDL when the one they are currently using is going to be retired by NetSuite (see Support for Existing WSDL Versions for details). If you do plan to upgrade the WSDL you are using, NetSuite strongly recommends that you upgrade to the latest available version to prolong the life cycle for your integrated application.
Important: You are not required to upgrade your WSDL when your NetSuite account is upgraded. However, when your account is upgraded, you should re-test all existing integrations to ensure they work against the latest version of NetSuite (see Testing After a NetSuite Upgrade for details).

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

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

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

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

Testing After a WSDL Upgrade

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

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

Testing After a NetSuite Upgrade

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

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

Note: As of 2018.1, all customers access the UI for their Release Preview account from https://system.netsuite.com. Initially, only users with an Administrator role have access to the Release Preview account. For more information, see the help topic The Release Preview Account.

- https://webservices.netsuite.com/wsd/v2019_2_0/netsuite.wsdl
NetSuite WSDL Versioning

- https://webservices.netsuite.com/wsdl/v2019_1_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2018_2_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2018_1_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2017_2_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2017_1_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2016_2_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2016_1_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2015_2_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2015_1_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2014_2_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2014_1_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2013_2_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2013_1_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2012_2_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2012_1_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2011_2_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2011_1_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2010_2_0/netsuite.wsdl
- https://webservices.netsuite.com/wsdl/v2010_1_0/netsuite.wsdl

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

**Working with the Araxis Merge Diff File**

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

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

For more information on using the Araxis diff file to review NetSuite WSDL changes, see the help topic *Araxis Merge Diff File for the 2019.2 Endpoint*, in the 2019.2 Release Notes.

**Support for Existing WSDL Versions**

At any time, the six latest endpoints are supported. For example, with the 2019.2 endpoint generally available, the following WSDL versions are supported:

- 2019.2
- 2019.1
- 2018.2
NetSuite WSDL Versioning

- 2018.1
- 2017.2
- 2017.1

If you use an older version, NetSuite recommends that you upgrade to the latest WSDL version.

NetSuite also makes the eight additional endpoint versions from the previous four years available without support. This means that in addition to the fully supported endpoints listed above, the following eight endpoints also remain available:

- 2016.2
- 2016.1
- 2015.2
- 2015.1
- 2014.2
- 2014.1
- 2013.2
- 2013.1

⚠️ Important: Starting from NetSuite version 2019.1, endpoints that are 15 and more versions older will be retired.

The release of new endpoints is announced in the Integration Newsletter. After your NetSuite implementation is upgraded to a newer version, retired endpoints are not available anymore.

Consider the following information when you are planning to upgrade to a newer endpoint.

- When your account is upgraded to a new version, support stops for the last endpoint on the list of supported endpoints, and the last endpoint on the list of available endpoints is retired.
- If you do not upgrade to a newer endpoint when your WSDL is retired, your integrations will cease to work because the endpoint for that version will not be available.

**SOAP Web Services Performance Optimization**

Consider the following guidelines to optimize SOAP web services performance:

- Use a Web Services Only form to minimize time to retrieve field values, included sourced fields and formula fields. For more information, see Forms in SOAP Web Services.
- Advanced Search allows reference to an existing saved search. For more information, see Reference Existing Saved Searches.
- Use context API to run minimal user event scripts for real time integration. For more information, see the help topic Specifying SOAP Web Services Context.
- Disable SuiteScript and workflow triggers for historical data imports. For more information, see runServerSuiteScriptAndWorkflowTriggers.
- Use list operations when possible. This approach works with multiple record types, for example, addList Customer then Sales Order. The ideal list size is around 20 transactions, 50 entities or items, and 100 custom records. For more information, see SOAP Web Services List Operations.
- Use asynchronous operations whenever possible. For more information, see SOAP Web Services Processing.
To benefit from increased concurrency, use token-based authentication. For SOAP web services that use token-based authentication (TBA) and for RESTlets, there are no per-user limits for concurrent requests, if the concurrency governance is enabled. For more information, see Web Services and RESTlet Concurrency Governance. This topic also contains a decision chart to help you decide whether you need more SuiteCloud Plus licenses to achieve optimal results.

- Optimize system notes logging.
  - Disable System Notes for custom fields and for custom record types. (SOAP preference). For more information, see disableSystemNotesForCustomFields.
  - Log System Notes on update only (general preference). For more information, see the help topic Setting General Account Preferences.

SOAP Web Services Governance Overview

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

- Requests are monitored and controlled to ensure that the user experience is not excessively impacted.
- The burden of heavy SOAP web services users is not shared among all users.

NetSuite SOAP web services governance includes:

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

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

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

Understanding Record Limiting

Limits exist for the following:

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

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

Synchronous Operations

<table>
<thead>
<tr>
<th>Operation or parameter (on a per request basis)</th>
<th>Minimum Record Count</th>
<th>Maximum Record Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>addList</td>
<td></td>
<td>200</td>
</tr>
<tr>
<td>deleteList</td>
<td></td>
<td>200</td>
</tr>
<tr>
<td>getList</td>
<td></td>
<td>1000</td>
</tr>
</tbody>
</table>
Asynchronous Operations

<table>
<thead>
<tr>
<th>Operation or parameter (on a per request basis)</th>
<th>Minimum Record Count</th>
<th>Record Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>asyncAddList</td>
<td>400</td>
<td></td>
</tr>
<tr>
<td>asyncGetList</td>
<td>2000</td>
<td></td>
</tr>
<tr>
<td>asyncUpdateList</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>asyncUpsertList</td>
<td>200</td>
<td></td>
</tr>
<tr>
<td>asyncDeleteList</td>
<td>400</td>
<td></td>
</tr>
<tr>
<td>pageSize parameter of searchPreferences complex type, when used in asynchronous searches</td>
<td>5</td>
<td>2000</td>
</tr>
</tbody>
</table>

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

Understanding Request Limiting

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

Understanding Governance Errors

The following faults are thrown because of other governance violations.

- **ExceededRecordCountFault** – thrown if a request exceeds the allowed record count.
- **ExceededRequestLimitFault** – thrown if the allowed number of concurrent requests is exceeded.
- **ExceededRequestSizeFault** – thrown if a request exceeds 100M.

For more information on exceptions, refer to SOAP Web Services Error Handling and Error Codes.
For information about concurrency governance, see Web Services and RESTlet Concurrency Governance.

Web Services and RESTlet Concurrency Governance

View a video overview about concurrency governance.
View the video about the latest enhancements to concurrency governance.

In the past, concurrency for web services and RESTlets was governed separately per user and authentication method. As of 2017.2, web services and RESTlet concurrency also began to be governed per account. The account governance limit applies to the combined total of web services and RESTlet requests.

For a visual overview of the governance model, see the Concurrency Governance Cheat Sheet PDF file.

For more details, see each of the topics listed here:
- Concurrency Governance Limits Based on Service Tiers and SuiteCloud Plus Licenses
- Concurrency Governance for Internal Applications
- Effects of Authentication Method on Concurrency Governance
- Account Concurrency Monitoring Resources
- Testing Your Concurrency Usage
- Errors Related to Concurrency Violations
- Recommended Actions
- Retrying Failed Web Services Requests
- Sample Scenario
- Frequently Asked Questions

Concurrency Governance Limits Based on Service Tiers and SuiteCloud Plus Licenses

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

<table>
<thead>
<tr>
<th>Service Tier</th>
<th>Account Base Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared, 3</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>1, 1+, 0</td>
<td>15</td>
</tr>
</tbody>
</table>

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

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

The account concurrency limit for your account is displayed on the Integration Governance page. You can access this page at Setup > Integration > Integration Management > Integration Governance.

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

To help you decide whether you need more SuiteCloud Plus licenses, see the following decision tree.
Concurrency Limit per Integration

Concurrency limit per integration allows you to allocate part of your account’s concurrent requests limit to specific integrations.

You can use concurrency limit per integration to ensure that:

- The specific integration to which this limit is allocated has the required bandwidth to function.
- The specific integration to which this limit is allocated does not use a too big part of your limit, blocking your other applications from working fluently.
To see and manage your available concurrency limit, go to Setup > Integration > Integration Management > Integration Governance.

Under Concurrency Usage you can view the account concurrency limit and the unallocated concurrency limit.

**Note:** The minimum unallocated concurrency limit value is one.

Under Concurrency Limits per Integration, you can see a list with all integration records, their state and the concurrency limit that has been allocated to them. If you are the owner of the integration, you can also see the Application ID.

When allocating part of your account's concurrency limit to one application, you reduce the available concurrency limit for other integrations, which do not have a specific limit.

**Important:** You should only use this feature if you have a good reason to do so. For example, you may have an external application that you do not fully control, and which occasionally sends a high number of concurrent requests. This sudden increase in concurrent requests can cause issues to other integrations. With this feature, you can now limit the number of concurrent requests allowed for this application to protect the functionality of the other integrations. It is a best practice to have one account concurrency limit, without limits for each integration.

To manage the concurrency limit per integration, click on the name of the integration to open the integration record.

Here you can enter the value for the concurrency limit you want to allocate in the Concurrency Limit field. In the MAX Concurrency Limit field, you can see the unallocated concurrency limit that is available to allocate.

**Note:** The value of the MAX Concurrency Limit field is always one less than the total unallocated concurrency limit of the account.

If your subscription for company Service Tier or the number of SuiteCloud Plus licenses expires, your account concurrency limit will reset to the default value. If the account concurrency limit decreases by that to a value lower than the sum of the allocated concurrency limit per integration, the administrator of the account receives a notification per email. Also, the first user to access Integration Governance sees a warning text on the top of the page.

### Concurrency Governance for Internal Applications

Certain internal applications are excluded from the scope of concurrency governance. Requests from the specified internal applications do not count towards your concurrency limit.

Additionally, requests from these applications are not included in the usage data displayed on the APM web services performance dashboard, the APM Concurrency Monitor tool, and the Integration Governance page. For information about these tools, see Account Concurrency Monitoring Resources.

Requests from these applications are not included in the usage data displayed on the Integration Governance page. For information about ways to monitor your concurrency usage, see Account Concurrency Monitoring Resources.

The following internal applications and SuiteApps are excluded from the scope of concurrency governance:

- Bronto – See the help topic Bronto Integration.
- OpenAir
Effects of Authentication Method on Concurrency Governance

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

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

_Important:_ The list of excluded internal applications is subject to change.

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

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

Account Concurrency Monitoring Resources

To find information about your account concurrency limits and concurrent usage, check the following resources:

- The Integration Governance page, available at Setup > Integration > Integration Management > Integration Governance displays information about whether account concurrency governance is enabled in your account, and also contains information about the account concurrency limit and the number and ratio of total and rejected requests.
- The SOAP Web Services Execution Log and SOAP Web Services Usage Log contain information about the requests that were rejected due to a violation of concurrency governance limits. For information about accessing the logs, see SOAP Web Services Execution Log and Using the SOAP Web Services Usage Log.
- The results of a web services operations search also contain information about web services operations that are rejected due to concurrency violation. For more information, see Web Services Operations Search Type.
- The Application Performance Management (APM) SuiteApp includes a web services performance dashboard. You can use this dashboard to monitor web services record processing and the status of web services operations, and to find out details about the operations that were rejected. For more information, see the help topic Application Performance Management (APM). For information about monitoring web services operations using APM, see the help topic Analyzing Web Services Performance.
The Application Performance Management (APM) SuiteApp also includes a Concurrency Monitor tool. You can use the Concurrency Monitor tool to monitor web services and RESTlet integrations with respect to your account concurrency limits. This dashboard enables you to view and optimize the scheduling of your integrations so that you can optimize the usage of your SuiteCloud plus licenses. The dashboard also shows you your exceeded concurrency data to help you decide if you need to add SuiteCloud Plus licenses to enhance your performance. For more information, see the help topic Application Performance Management (APM). For more information about the Concurrency Monitor tool, see the help topic Monitoring Web Services and RESTlet Concurrency.

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

Testing Your Concurrency Usage

In sandbox accounts, you can turn the account concurrency governance on and off to test how your usage fits with your account concurrency limit. You can enable or disable the concurrency governance by selecting or clearing the Web Services and RESTlet Account Governance Enabled box on the Integration Governance page.

For more information, see Using Integration Records in Sandbox Accounts.

Errors Related to Concurrency Violations

RESTlet and web services requests that are rejected because they exceed the concurrency governance limit result in one of the following server responses.

For RESTlet requests, the following errors occur:

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

For web services requests, the following SOAP faults occur:

<table>
<thead>
<tr>
<th>Authentication Type</th>
<th>SOAP Fault</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request-level credentials</td>
<td>ExceededRequestLimitFault</td>
<td>WS_CONCUR_SESSION_DISALLWD</td>
</tr>
<tr>
<td>Login and logout operations</td>
<td>ExceededConcurrentRequestLimitFault</td>
<td>WS_REQUEST_BLOCKED</td>
</tr>
<tr>
<td>TBA</td>
<td>ExceededConcurrentRequestLimitFault</td>
<td>WS_REQUEST_BLOCKED</td>
</tr>
</tbody>
</table>

Recommended Actions

Analyze the frequency and level of concurrency peaks, and consider rescheduling requests to be outside of regular peak times.

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

```java
int i = 0;
    int maxAttempts = 5; // try it 5 times, then fail for good
while (i < maxAttempts) {...
```
```java
{
    response = doWSCall();
    isSuccess = response.getIsSuccess();
    errorMsg = response.getErrorMsg();

    if (isSuccess == false && (errorMsg == WS_CONCUR_SESSION_DISALLWD || errorMsg == WS_REQUEST_BLOCKED))
    {
        wait();
        i++; // try again
    }
    else
    {
        break; // end the cycle
    }
}
```

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

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

## Retrying Failed Web Services Requests

If your client application receives the WS_CONCUR_SESSION_DISALLWD response, or you do not receive any response, you should retry sending the request after a certain period.

Consider the following best practices when implementing retry logic:

- You should send the first retry attempt after a certain delay. If more attempts are needed, you should increase the delay between attempts.
- Avoid the creation of artificial concurrency peaks due to synchronized retry attempts from different threads or applications. There is no specific recommended length of delay between retry attempts to avoid synchronization.
- If retry attempts are needed for interactive applications, you should execute requests asynchronously so there is no need for repeated user action.

## Sample Scenario

Consider Company ABC that is on a shared service tier with one SuiteCloud Plus license. User A is the designated concurrent web services user. The company makes 10 web services requests concurrently. User A uses request-level credentials to make web services requests. The company also has one RESTlet that is reached concurrently five times. For RESTlet requests, token-based authentication (TBA) is used. User B generates one web services request using request-level credentials. All of these requests occur in parallel.

The account-level concurrency limit of this customer is 15 (5 + 10). If they generate 5 RESTlet requests using TBA, 10 web services requests with request-level credentials from User A, and one web services request with request-level credentials from User B, an error will be thrown for one of the requests.

This error occurs when they generate one request over their account-level limit. The first 15 requests will succeed, and the 16\textsuperscript{th} concurrent request will result in an error. The error can occur for any of the requests. If Company ABC does not want to use a dedicated concurrent web services user, they can re-
implement their web services integrations to use TBA instead of request-level credentials. The company still needs the SuiteCloud Plus license to get support for 15 web services concurrent requests.

**Frequently Asked Questions**

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

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

**Will my existing integrations break due to these changes?**

If the concurrency limits are exceeded, your integration might encounter the error codes listed in Effects of Authentication Method on Concurrency Governance. You must modify your integration to handle these exceptions and you must add logic to retry the requests.

**Will this affect my sandbox account governance?**

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

**Can I monitor my account’s concurrency usage level?**

You can check your account limit on the Integration Governance page, at Setup > Integration > Integration Governance.

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

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

**How does this impact the SuiteCloud Plus license?**

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

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

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

We recommend that you use TBA. This eliminates the need to designate a concurrent web service user as one token can be used in more concurrent requests up to the account limit.
How does the governance change affect the usage of the Concurrent Web Service User option on the user record?

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

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

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

Vocabulary

You should be familiar with the following terms before implementing SOAP web services integration technology.

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

These sections provide the basic information you need to get started with SOAP web services.

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

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

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

### SOAP Web Services Quick Start

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

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

Note the following:

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

### Enabling the SOAP Web Services Feature

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

**To enable the SOAP web services feature:**

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

Note that after you have enabled SOAP web services in your NetSuite account, you should set your SOAP web services preferences. See SOAP Web Services Preferences for more details.
Enabling the Show Internal IDs Preference

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

**To enable the Show Internal IDs preference:**

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

When this preference is enabled:

- You can view the internal ID for a field by clicking on that field's label to open the field level help popup window. The internal ID is displayed in the lower right corner of this window.
- You can view the internal ID for a record or a custom field in an Internal ID column that displays on a list or search results page for that type of record or custom field.
- You can view the internal IDs for workflow actions on the workflow context panel, workflow action group window, and sublist action group window. You can also view internal IDs in workflow execution logs.

Changes to this preference affect only the current user.

For more details, see the help topic [Showing Record and Field IDs in Your Account](https://www.netsuite.com/nc/ref/ncdev/ncdev13.htm).

Building an Application with Microsoft .NET

This section provides details on how to use the Microsoft .NET platform to build a SOAP web services application. You can also see [Downloading Sample Applications](https://www.netsuite.com/nc/ref/ncdev/ncdev13.htm) for a .NET sample currently available on the NetSuite Developer Portal.

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

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

**To use Microsoft .NET with NetSuite SOAP web services:**

1. Install the Microsoft .NET framework.
   - Install Microsoft Visual Studio .NET, version 2003 or higher, which includes the .NET framework SDK (RECOMMENDED), or the Microsoft .NET Framework SDK version 1.1 (NOT recommended).
2. Use the .NET framework SDK to automatically generate the client proxy code
   - **If using Visual Studio .NET (RECOMMENDED):**
     1. Start Microsoft Visual Studio .NET.
     2. Create a new project and choose a template, for example, Console Application.
3. After the project has been created, add a Web reference.
   The Add Web Reference option may be available in the Project menu. If not, select Add Service Reference, click the Advanced button, and click the Add Web Reference button.

4. In the Add Web Reference dialog, enter the SOAP WSDL URL and click the Go icon.
   A URL for the latest SOAP WSDL is: https://webservices.netsuite.com/wsdl/v2019_2_0/netsuite.wsdl

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

   **Note:** Although the material in this guide pertains to the latest NetSuite WSDL, we continue to support endpoints released prior to this version.

5. Visual Studio inspects the WSDL and displays a summary of the available operations. If security warnings are displayed, click Yes as many times as necessary to continue this process.

6. After the summary is displayed, click the Add Reference button to generate the classes.
   When this process is complete, com.netsuite.webservices is listed under Web References in the Solution Explorer.

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

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

   1. Locate the wsdl.exe file under the Microsoft .NET Framework SDK installation and add it to your system path.

   2. Open a command prompt, and type the following to generate the proxy classes:

   ```
   C:\project>wsdl /language:<language> <url>
   ```
   
   Where <language> is your preferred language and <url> is the URL for the NetSuite WSDL.
   For example, generate the proxy classes in C# as follows:

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

   A C# file called NetSuiteService.cs is generated.

   3. Compile the source code for your proxy into a .NET Assembly using the .NET Framework C# compiler or any other supported compiler.

   4. Locate the csc.exe file under the Microsoft .NET Framework SDK installation and add it to your system path.

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

   3. Implement your application by writing your business logic using the generated .NET proxy classes.
Building an Application with Java using Apache Axis

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

To use the Apache Axis framework with NetSuite SOAP web services:

1. Install the Java 2 Platform.
   Download and install the Java 2 Platform, Standard Edition, version 1.4 or higher from https://www.oracle.com/technetwork/java/javase/index-jsp-138567.html. Ensure that the executables are available through the system path.

2. Install Apache Axis.

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

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

3. Install Apache Ant (Optional).

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

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

4. Configure Java to generate unreferenced types.

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

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

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

Using Apache Axis from the command line

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

```bash
java -cp <classpath> org.apache.axis.wsdl.WSDL2Java -a -w -O 120000 <url>
```

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

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

```
>set CP=%AXIS_HOME%\lib\axis.jar;%AXIS_HOME%\lib\jaxrpc.jar;%AXIS_HOME%\lib\commons-discovery.jar;%AXIS_HOME%\lib\wsdl14.jar;%AXIS_HOME%\lib\saaj.jar;
>java -cp %CP% org.apache.axis.wsdl.WSDL2Java https://webservices.netsuite.com/wsd/2017_1/0/netsuite.wsd
```

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

Using the Apache Axis task in Apache Ant

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

```xml
<target name="generate.interfaces" description="Generates client interfaces using Axis">
  <echo>Generating client interfaces using Apache Axis</echo>
</target>
```
6. Ensure that the Ant executables are available on the system path. Run the Ant task as follows:

```
ant generate.interfaces
```

7. Implement your application by writing your business logic using the generated axis proxy classes.

1. Locate the NetSuite service.

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

2. Enable support for multiple cookie management.

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

3. Get the NetSuite port.

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

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

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

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

```
Customer cust = new Customer();
cust.setEntityID( "XYZ Inc" );
WriteResponse response = port.add( cust );
```

6. Log out to invalidate the current session.

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

### Downloading Sample Applications

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

### Capturing SOAP

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

SOAP Web Services Development Considerations

This section covers the following topics:

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

Development Considerations Overview

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

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

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

1. A successful login operation must be performed to obtain a session to perform any subsequent operations. (See the documentation on the login operation and on SOAP web services Authentication for SOAP Web Services for more details.)
2. Some add operations are done in a tandem, such as loading an existing sales order to gain the context and then adding a new item fulfillment record. The loading of the sales order (using the initialize API) is the first operation; adding the item fulfillment record is the second.
3. Restrictions and requirements on custom forms are honored in SOAP web services. Therefore, a SOAP application's attempt to set a field that is hidden on a form results in a permission error. Likewise, as in the UI, a SOAP application must set fields that are required on the form it is using.

The combination of permission adherence and similar behavioral patterns between SOAP web services and the NetSuite UI provides a consistent and predictable platform for developers.
**Important:** When you are unsure of how to achieve something with SOAP web services, try observing how it is done in the UI, then replicate it programmatically.

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

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

### NetSuite Features in SOAP Web Services

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

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

### SOAP Web Services Reliability Considerations

You may encounter issues when a SOAP web services operation does not complete or the client does not receive a response. This topic provides the most common causes of this issue and provides best practices to help you avoid them.

#### Reasons for SOAP Web Services Operation Failures

- **Network connection errors at any point of the connection or minor network disruptions and timeouts.** Such disruptions can occur anytime. Note that if the server receives the request, the processing of operations continues, even if a network disruption occurs later. In such a case, it is possible that the client does not receive a response from the server, so it is not clear for the client whether the operation completed successfully or failed.
- **Error in the NetSuite infrastructure.** For example, an application server might crash.
- **Client error.** Due to such errors, the client might not receive a response from the server, and cannot assess whether the operation succeeded or failed.
- **Exceeding a timeout:**
  - Web services sessions automatically time out after 20 minutes of inactivity, requiring a login to resume activity.
  - If a web services request takes more than 15 minutes to complete, it automatically times out.

**Important:** List and search requests that take longer than 15 minutes to process are terminated. For more information about handling long-running list and search operations, see [Handling of Lengthy Requests](#).

### Recommended Actions

The following sections provide recommended actions for troubleshooting, and best practices for avoiding issues with operation failures.
Client Implementation Considerations and Troubleshooting

- For long-running SOAP web services operations, consider using asynchronous request processing. For more information, see Synchronous Versus Asynchronous Request Processing. Note that the asyncSearch operation can also time out, and this timeout does not necessarily correspond to the general operation timeout.
- Use the SOAP web services operations search to search for and analyze information about failing SOAP web services requests and operations. For more information, see Web Services Operations Search Type.
- Check the SOAP web services usage log for information about the operation and the status of the request. For more information, see Using the SOAP Web Services Usage Log.
- When an afterSubmit script runs after a SOAP web services add or update request and the script fails, the records are still created or updated in NetSuite. Using the internal ID of the record, available in the SOAP response, you can avoid creating duplicate records. If the SOAP response indicates that the records have been saved, you should not send the SOAP web services request again. For more information, see SOAP Web Services Warnings, Errors, and Faults.

Synchronous Client Implementation

- To prevent duplicate records and to make sure that all records are correctly added, use external IDs and the upsert and upsertList operations to update records in NetSuite.

  **Note:** External IDs originally serve to identify records in a third party system. If you cannot use such an identification, generate a random but unique external ID, for example, a UUID, which you can then use to create records in NetSuite.

- You must implement your business logic in a way that your update operation is idempotent.
  - Idempotency is also necessary for working with sublists. When you work with sublists in SOAP web services, the approach for updating sublist lines varies depending on whether you are working with keyed sublists or non-keyed sublists. Keyed sublists allow you to set the replaceAll attribute to false, to update only the specific lines you submit in your requests. For non-keyed sublists, the replaceAll attribute is ignored, and you must interact with the sublist as a whole. For detailed information, see Updating Sublists in SOAP Web Services.

Implementing a Retry Logic

You must implement a retry logic to resend SOAP web services requests that do not complete. Using a retry mechanism is essential for the following reasons:

- To make sure that requests that fail due to timeout issues are executed again.

  **Note:** If a request times out and you send a retry request, it is possible that the response contains fewer records than the first response.

- To make sure that requests rejected due to concurrency violations are executed again. For information about concurrent sessions, see Web Services and RESTlet Concurrency Governance.

- To make sure that you do not violate SOAP web services session management timeouts and limits. For more information, see Session Management for SOAP Web Services.

See the sample retry code below for an example.

**Sample Retry Code**

```java
int i = 0;
int maxAttempts = 5; // try it 5 times, then fail for good
```
Effects of Account Configuration in SOAP Web Services

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

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

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

Effects of Accounts' Host Data Center on URLs

Your integration must incorporate logic that dynamically determines the correct URL.

For more details about dynamic discovery of URLs, see the help topic Dynamic Discovery of URLs for SOAP Web Services and RESTlet Clients.

Important: As of the 2019.1 SOAP web services endpoint, you must use account-specific domains with SOAP web services. You can use the SOAP getDataCenterUrls operation to obtain the correct domain. Or, go to Setup > Company > Setup Tasks > Company Information in the NetSuite UI. Your domains are listed on the Company URLs subtab. While earlier endpoint versions still support data center-specific domains, it is recommended that you use account-specific domains as much as possible.

The following methods exist for determining the correct URL:

- In the 2012.2 and later endpoints, you can use the getDataCenterUrls operation. This operation is not supported in earlier endpoints.
- You can use the REST roles service provided by NetSuite.
  - For general information about this service, see The REST roles Service.
  - For example code that uses this REST service, see:
    - Java REST Example for SOAP Web Services Domain Discovery
The NetSuite Developer Resources sample applications, which also incorporate RESTful code for dynamic discovery of SOAP web services domain. These samples are available at: http://www.netsuite.com/portal/developers/resources/suitetalk-sample-applications.shtml

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

### SOAP Web Services URLs for Sandbox and Release Preview Accounts

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

**Note:** As of 2018.1, all customers access the UI for their Release Preview account from https://system.netsuite.com. Initially, only users with an Administrator role have access to the Release Preview account. For more information, see the help topic The Release Preview Account.

### Java REST Example for SOAP Web Services Domain Discovery

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

**Important:** NetSuite hosts customer accounts in multiple data centers. For that reason, the correct URL for SOAP web services access varies depending on the data center hosting the account.

For details about dynamic discovery of URLs, see Effects of Accounts' Host Data Center on URLs and Dynamic Discovery of URLs for SOAP Web Services and RESTlet Clients.

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

**Important:** As of the 2019.1 SOAP web services endpoint, you must use account-specific domains with web services. You can use the SOAP getDataCenterUrls operation to obtain the correct domain. Or, go to Setup > Company > Setup Tasks > Company Information in the NetSuite UI. Your domains are listed on the Company URLs subtab.

### Sample Java Call to the REST roles Service

```java
package com.netsuite.prototypes;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import org.json.simple.parser.ParseException;
import javax.net.ssl.HttpsURLConnection;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;
public class GetRESTDomain
```
static class URLTriplet
{
    String restDomain;
    String systemDomain;
    String webservicesDomain;
}

private BufferedReader _br = null;
public GetRESTDomain()
{
    _br = new BufferedReader(new InputStreamReader(System.in));
}

/** *
 * Retrieve domains of a NetSuite account through RESTful WS call.**/

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

        for (int i=0 ; i<jsonArr.size() ; i++)
        {
            JSONObject json = (JSONObject) jsonArr.get(i);
            // Validate JSON object if (json.containsKey("account") && json.containsKey("dataCenterURLs"))
            
                JSONObject jsonAccount = (JSONObject) json.get("account");
                JSONObject jsonDataCenters = (JSONObject) json.get("dataCenterURLs");

                // Validate account and retrieve domains if
                (jsonAccount.get("internalId").toString().equalsIgnoreCase(nsAccount))
                {
                    urls.restDomain = jsonDataCenters.get("restDomain").toString();
                    urls.webservicesDomain = jsonDataCenters.get("webservicesDomain").toString();
                    urls.systemDomain = jsonDataCenters.get("systemDomain").toString();
                    break;
                }
            
        }
    }

    return urls;
}
public static void main(String[] args) throws IOException
{
    GetRESTDomain ns = new GetRESTDomain();
    while (true)
    {
        System.out.println("GET REST DOMAIN URL");
        System.out.print("Account      : ");
        String account = ns._br.readLine().trim();
        if (account.isEmpty())
        {
            break;
        }
        System.out.print("Email address: ");
        String email = ns._br.readLine().trim();
        System.out.print("Password     : ");
        String password = ns._br.readLine().trim();
        try
        {
            URLTriplet urls = ns.getDataCenterUrls(account, email, password);
            System.out.println("\nREST domain URL        : " + urls.restDomain + "\n");
            System.out.println("Web services domain URL: " + urls.webservicesDomain + "\n");
            System.out.println("System domain URL      : " + urls.systemDomain + "\n");
        }
        catch (IOException e)
        {
            e.printStackTrace();
        }
        catch (ParseException e)
        {
            e.printStackTrace();
        }
    }
    System.out.print("\nPress any key to exit...");
    ns._br.readLine().trim();
}
}

Sample AXIS Client

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

Enumerations and Special Characters

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

Image References in SOAP Web Services

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

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

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

Company-Wide Preferences

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

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

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

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

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

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

Disable Mandatory Custom Field Validation

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

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

This preference can be overridden at the request level. For details, see disableMandatoryCustomFieldValidation.
Run Server SuiteScript and Trigger Workflows

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

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

If you disable this option, you do both of the following:
- Disable server SuiteScripts that otherwise would have been triggered by SOAP web services requests.
- Disable workflows that otherwise would have been triggered by SOAP web services requests.

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

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

Require Approval During Auto-Installation of Integration

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

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

Search Page Size

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

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

Treat Warnings as Errors

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

This preference can be overridden at the request level. For details, see warningAsError.
Use Conditional Defaults on Add

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

For details on this preference, see Resetting Default Behavior.

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

Use Conditional Defaults on Update

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

For details on this preference, see Resetting Default Behavior.

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

One-Time Preferences

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

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

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

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

Request-Level Preferences

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

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

Note that all of these preferences take boolean values.

The available preferences are:

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

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

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

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

disableSystemNotesForCustomFields

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

This preference is available at the request level only.

In NetSuite, logging system notes can also be controlled by checking the Log System Notes on Update Only box under Setup > Company > Preferences > General Preferences. The SOAP web services preference does not override the UI preference when the UI preference is set to true. This means that if you set the Log System Notes on Update Only preference in the UI to true, while the disableSystemNotesForCustomFields SOAP web services preference to false, only updates will be logged for custom fields. For more information about the UI preference, see the help topic Setting General Account Preferences.

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

ignoreReadOnlyFields

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

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

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

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

### runServerSuiteScriptAndWorkflowTriggers

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

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

```xml
<runServerSuiteScriptAndWorkflowTriggers>true</runServerSuiteScriptAndWorkflowTriggers>
```

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

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

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

Also see Basic Search Behavior with User Event Scripts and Workflows about the effects of this preference.

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

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

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

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

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

### useConditionalDefaultsOnAdd

This preference is available only for the 2009.2 endpoint and earlier. Note that these older endpoints are no longer supported. For details, see Support Has Ended for SOAP Web Services 2009.2 and Earlier Endpoints.
For details on the behavior of the preference, see Resetting Default Behavior.

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

useConditionalDefaultsOnUpdate

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

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

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

warningAsError

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

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

bodyFieldsOnly

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

Leaving this preference set to its default value, which is true, can significantly improve performance. If the preference is set to false, a basic search triggers user event scripts and workflows.

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

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

pageSize

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

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

<table>
<thead>
<tr>
<th>Search Type</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous</td>
<td>5</td>
<td>1,000</td>
</tr>
<tr>
<td>Asynchronous</td>
<td>5</td>
<td>2,000</td>
</tr>
</tbody>
</table>
If you enter an invalid number, the system may return an error code reading `WS_INVALID_SEARCH_PAGE_SIZE`.

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

**returnSearchColumns**

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

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

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

**Sample Code**

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

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

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

### C#

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

```csharp
// Set up request-level preferences as a SOAP header
Preferences prefs = new Preferences();
_service.preferences = prefs;

// Set preference to specify that warnings should be treated as errors
prefs.warningAsErrorSpecified = true;
prefs.warningAsError = false;
```

In this example, search preferences are set.

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

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

### Java

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

```java
// Set up request-level preferences as a SOAP header
Preferences prefs = new Preferences();
_service.preferences = prefs;

// Set preference to specify that warnings should be treated as errors
prefs.warningAsErrorSpecified = true;
prefs.warningAsError = false;
```

In this example, search preferences are set.

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

_service.searchPreferences.pageSize = 20;
_service.searchPreferences.pageSizeSpecified = true;
SearchResult response = _service.search( custSearch );
```
private void setPreferences() throws SOAPException {
    stub = (NetSuiteBindingStub) _port;
    stub.clearHeaders();
    SOAPHeaderElement prefHeader =
        new SOAPHeaderElement("urn:messages_2017_1.platform.webservices.net_suite.com", "preferences");
    Preferences prefs = new Preferences();
    prefs.setWarningAsError(new Boolean(false));
    prefs.setUseConditionalDefaultsOnAdd(new Boolean(false));
    prefs.setUseConditionalDefaultsOnUpdate(new Boolean(false));
    prefs.setDisableMandatoryCustomFieldValidation(new Boolean(true));
    prefs.setDisableSystemNotesForCustomFields(new Boolean(true));
    prefs.setIgnoreReadOnlyFields(new Boolean(true));
    prefHeader.setObjectValue(prefs);
    stub.setHeader(prefHeader);
}

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

**SOAP Request**

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

```xml
<soap:Header>
<platformMsgs:preferences>
    <platformMsgs:warningAsError>true</platformMsgs:warningAsError>
    <platformMsgs:useConditionalDefaultsOnAdd>true</platformMsgs:useConditionalDefaultsOnAdd>
    <platformMsgs:useConditionalDefaultsOnUpdate>true</platformMsgs:useConditionalDefaultsOnUpdate>
    <platformMsgs:disableMandatoryCustomFieldValidation>true</platformMsgs:disableMandatoryCustomFieldValidation>
    <platformMsgs:disableSystemNotesForCustomFields>true
</platformMsgs:preferences>
</soap:Header>
```

**Resetting Default Behavior**

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

Within the NetSuite UI, there are three types of default behaviors that may be associated with any particular record.
- Record fields can be automatically populated with default values.
- Records can have related fields that are automatically populated with default values when an initial value is entered. These fields are populated depending on the condition of the initial field.
- Records can be populated with a calculated value depending on the values set in a particular field.

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

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

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

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

**Example**

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

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

![Primary Information](image)

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

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

**To display internal ID values:**

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

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

Changes to this preference affect only the current user.

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

Caching Behavior in SOAP Web Services

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

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

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

As of 2015.2, all external applications that send SOAP web services requests to your NetSuite account are represented by one or more integration records. You can use integration records to manage applications, view activity logs, and more.

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

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

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

Integration Record Overview

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

For more details, see the following sections:

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

Benefits of the Integration Record

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

- **View details about each application** – Each integration record shows details such as the application’s name and description.

- **Block an application** – You can use an integration record to block SOAP web services requests that come from the corresponding application. However, if you are using the 2015.1 or an earlier endpoint, be aware that some requests may be handled by the Default Web Services Integrations record, which has its own configuration options. For details, see [Blocking SOAP Web Services Requests](#).

- **Select permitted authentication methods for an application** – When you create an integration record, you can permit the application to authenticate through user credentials, token-based authentication, or both. Note that when a request authenticates through user credentials, the request must also include an application ID, if you want the application's requests to be tracked by the appropriate integration record. (For details, see Application IDs.) Additionally, be aware that the permitted authentication methods vary depending on which endpoint the request uses. For details, see [Endpoint Requirements](#).

- **Distribute an integration record** – You can bundle an integration record and make it available for installation in other NetSuite accounts. You can also configure a record so that it can be automatically installed in other accounts. For details, see [Distributing Integration Records](#).

- **View an execution log specific to each application** – When you create and use an integration record for each application, that record's SOAP Web Services Execution Log lets you view only those requests sent by that application. If you are not using unique integration records for each application, you can view all requests together by using the log at Setup > Integration > SOAP Web Services Usage Log.

- **View a list of all your applications** – You can view a list of all your applications at Setup > Integration > Manage Integrations.

### Application IDs

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

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

Use the following guidelines:

- Requests that authenticate through user credentials are required to include an application ID, if the request uses the 2015.2 endpoint or later. (For details, see [Endpoint Requirements](#).) If the request uses the 2015.1 endpoint or earlier, an application ID is optional.

- Inbound SSO requests must include an application ID, if the requests use the 2015.2 endpoint or later. If the request uses the 2015.1 endpoint or earlier, an application ID is optional.

- Application IDs should not be included in outbound SSO calls, regardless of which endpoint is being used.

- If a request uses token-based authentication, you must not include an application ID, regardless of which endpoint you are using. If you include an application ID with a request that uses TBA, the request fails.

Although application IDs are considered identifiers and not authentication credentials, application IDs should nevertheless be considered confidential.
Important: Some customers may have application IDs that were provided to them by NetSuite Customer Support prior to 2015.2. These older IDs must not be used with the 2015.2 or a later WSDL. Older application IDs are permitted only in requests that use the 2015.1 WSDL or earlier. These requests are logged as part of the Default Web Services Integrations record’s SOAP Web Services Execution Log.

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

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

Endpoint Requirements

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

2015.1 and Earlier

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

- You have the option of letting some or all of your requests be tracked by a single integration record called Default Web Services Integrations. This record requires little maintenance, if any. The Default record was automatically created when your account was upgraded to 2015.2 or subsequently, when you enabled the Web Services feature. The behavior of this record is slightly different from the behavior of integration records that you create. For details on this record, see Default Web Services Integrations Record.
- Token-based authentication is not supported. If you want a request to be tracked by a specific integration record (other than the Default record), it must authenticate through user credentials and include an application ID. For more details about application IDs, see Application IDs.

2015.2 and Later

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

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

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

2016.2 and Later

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

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

For details about SOAP web services searches, Searching for SOAP Web Services Log Information.

Ownership of Integration Records

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

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

- Installed records can be installed as part of a bundle.
- In some cases, integration records can also be auto-installed. With this approach, a record is automatically installed when a request is sent that references the record’s application ID. Auto-installation is supported both for integration records that are configured to permit token-based authentication and authentication through user credentials.

**Important:** As of 2019.1, when calling the Issue Token endpoint, an Integration record is created and automatically installed in your account. The Require Approval during Auto-Installation of Integration preference affects whether this new record is automatically enabled. You can manage the preference at Setup > Integration > Integration Management > SOAP Web Services Preferences. If the box for the Require Approval during Auto-Installation of Integration preference is not checked (set to false) the State field on the new application is automatically set to Enabled, and all requests are permitted. However, if the box is checked (set to true) the State field on the new integration record is set to Waiting for Approval. In the latter case, you must manually edit the record and set the State to Enabled. Until you set the state to Enabled, all requests sent by that application are blocked.

Records that are installed through either of these methods are not fully editable, because they are considered to be owned by a different NetSuite account. On such records, you can make changes to only two fields: the Note field and the State field. All other fields on these records, including the permitted authentication options and the Description field, can be changed only by an authorized user in the account that owns the records. When the owner makes changes, they are pushed automatically to your account. These changes are not reflected in the system notes that appear in your account.

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

Adding an Integration Record

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

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

Creating an Integration Record

You can add an integration record to your account by auto-installing the record, or by installing a bundle that includes the record. However, sometimes you must create an integration record by filling out a form
Adding an Integration Record

in the UI. For example, if you have created an integration for your own use only, you should create an integration record by filling out the Integration form at Setup > Integration > Manage Integrations > New.

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

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

**Note:** Only administrators can complete this procedure.

**To manually create an integration record:**

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

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

If you selected the Token-based Authentication option, the updated page also shows the record's consumer key and secret.
Warning: For security reasons, the only time the Consumer Key and Consumer Secret values are displayed is on the confirmation page. After you leave this page, these values cannot be retrieved from the system. If you lose or forget these credentials, you must reset them to obtain new values. Treat these values as you would a password. Never share these credentials with unauthorized individuals and never send them by email.

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

Auto-Installation of Integration Records

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

Auto-installation can be used both with integration records that are configured to authenticate through user credentials or token-based authentication.

Important: As of 2019.1, when calling the Issue Token endpoint, an Integration record is created and automatically installed in your account. The Require Approval during Auto-Installation of Integration preference affects whether this new record is automatically enabled. You can manage the preference at Setup > Integration > Integration Management > SOAP Web Services Preferences. If the box for the Require Approval during Auto-Installation of Integration preference is not checked (set to false) the State field on the new application is automatically set to Enabled, and all requests are permitted. However, if the box is checked (set to true) the State field on the new integration record is set to Waiting for Approval. In the latter case, you must manually edit the record and set the State to Enabled. Until you set the state to Enabled, all requests sent by that application are blocked.

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

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

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

Understanding Auto-Installation

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

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

It is also possible to auto-install an integration record from your sandbox to your production account. Or, if an integration record was removed from a sandbox account during a sandbox refresh, you can restore it by auto-installing it.
In all of these cases, after a request is sent that includes the application ID, the integration record is automatically installed. However, SOAP web services requests from the application may or may not be automatically permitted. This behavior varies depending on the configuration of a preference, as described in Configuring the Require-Approval Preference.

Configuring the Require-Approval Preference

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

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

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

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

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

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

Bundle Installation of Integration Records

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

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

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

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

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

Application Details for Client Code

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

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

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

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

Requirements for Using an Application ID and User Credentials

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

- An integration record for the application must exist at Setup > Integration > Manage Integrations. On the integration record, the User Credentials option must be enabled. For help creating an integration record, see Creating an Integration Record.
- Your request must include, either explicitly or implicitly, the application ID that was generated when the integration record was created. If the integration was created in your NetSuite account, you can view the application ID on the integration record. If the integration record was created by a partner, in most cases you will obtain an updated integration from the partner that includes the application ID. With request-level credentials, the application ID must be included in the header of each request. For the login or ssoLogin request, you have an additional option, as described in Using Application ID with the Login and ssoLogin Operations.
  
  The request must use a Passport object, not a TokenPassport. For examples of how to include Passport data in a request, see Authentication for SOAP Web Services.

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

Application ID Code Samples (Request-Level Credentials)

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

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

The ApplicationInfo complex type is defined in the Messages XSD.

To add an application ID to an integration:

1. Create an ApplicationInfo class member. For example:

```csharp
namespace NetSuite
{
```
class SOAPApplication
{
    public NetSuiteService _service;
    private Preferences _prefs;
    private SearchPreferences _searchPreferences;
    private ApplicationInfo _appInfo;

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

private void setApplicationInfo()
{
    _appInfo = new ApplicationInfo();
    _service.applicationInfo = _appInfo;
    _appInfo.applicationId = "C4A2541D-16FB-4845-B8A2-8B8A34B0B61";
}

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

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

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

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

<soap:Header>
  <applicationInfo xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <applicationId>A7EE88F2-90AA-43A2-9948-29AA0E2DA61</applicationId>
  </applicationInfo>
  <passport xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <email xmlns="urn:core_2017_1.platform.webservices.netsuite.com">user@wsuser.com</email>
    <password xmlns="urn:core_2017_1.platform.webservices.netsuite.com">**********</password>
    <account xmlns="urn:core_2017_1.platform.webservices.netsuite.com">1013519</account>
    <role internalId="3" xmlns="urn:core_2017_1.platform.webservices.netsuite.com"/>
  </passport>
  <preferences xmlns="urn:messages_2017_1.platform.webservices.netsuite.com"/>
</soap:Header>

Using Application ID with the Login and ssoLogin Operations

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

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

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

Requirements for Using Token-Based Authentication

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

- General Setup Requirements
- Additional SOAP Web Services Requirements

General Setup Requirements

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

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

Additional SOAP Web Services Requirements

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

- The request must be made by using the 2015.2 WSDL or later.
- The request must use request-level credentials, not the login operation. For details on the difference, see Authentication for SOAP Web Services.
The request must use a TokenPassport object, not a Passport object. For help updating an integration to use a TokenPassport, see TokenPassport Complex Type and Updating an Integration to Send Token-Based Authentication Details.

Regenerating a Consumer Key and Secret

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

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

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

Note: Only administrators can complete this procedure.

To regenerate a consumer key and secret:

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

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

TokenPassport Complex Type

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

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

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

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

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

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

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

TokenPassportSignature

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

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

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

For example, suppose you have the following variables:

- **NetSuite account ID** – 1234567
- **Consumer key** – 71cc02b731f05895561ef0862d71553a3ac99498a947c3b7beaf4a1e4a29f7c4
- **Token** – 89e08d9767c5ac85b374415725567d05b54ecf0960ad2470894a52f741020d82
Application Details for Client Code

- **Nonce** – 6obMKq0tmY8yIVODeKdA1
- **Timestamp** – 1439829974

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

```
1234567&71cc02b73f105895561ef0862d71553a3ac99498a947c3b7b4ef4a49a29f7c454909e08d9767c5ac85b374415725557d05b54ecf8960ad2478894a52f741
```

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

For example, suppose you have the following variables:

- **Consumer secret** – 7278da58ca075fc336301a601203d10a58e948eda280f0618e25fcee1ef2abd
- **Token secret** – 060cd9ab3ffbe1e3d3918e90165fd273ab12acc764691046e2d29c7d7674c2

In this case, the key would be as follows:

```
7278da58ca075fc336301a601203d10a58e948eda280f0618e25fcee1ef2abd060cd9ab3ffbe1e3d3918e90165fd273ab12acc764691046e2d29c7d7674c2
```

3. Choose a **supported hash algorithm** to create an RFC 2104-compliant signature. Supported algorithms include HMAC-SHA1 and HMAC-SHA256.

4. Use the base string, the key, and the algorithm to create an RFC 2104-compliant signature. The signature must be encoded by Base64.

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

### Updating an Integration to Send Token-Based Authentication Details

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

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

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

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

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

1. Add logic for creating an RFC 2104-compliant signature. For example, you could add a method that has three arguments: a base string, a key, and an algorithm. Later, you could call this method by passing values specific to your request. (For details on the data needed to create the correct base string and key values, see [TokenPassportSignature](#).)

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

   ```java
   protected String computeShaHash(String baseString, String key, String algorithm) throws Exception {
   ```
if (!algorithm.equals("HmacSHA256") && !algorithm.equals("HmacSHA1")) algorithm = "HmacSHA1";
{
    byte[] bytes = key.getBytes();
    SecretKeySpec mySigningKey = new SecretKeySpec(bytes, algorithm);
    Mac messageAuthenticationCode = Mac.getInstance(algorithm);
    messageAuthenticationCode.init(mySigningKey);
    byte[] hash = messageAuthenticationCode.doFinal(baseString.getBytes());
    String result = new String(Base64.encodeBase64(hash, false));
    return result;
}
}

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

public String computeSignature(
    String account,
    String consumerKey,
    String consumerSecret,
    String token,
    String tokenSecret,
    String nonce,
    Long timeStamp,
    String signatureAlgorithm
) throws Exception
{
    String baseString = account + "&" + consumerKey + "&" + token + "&" + nonce + "&" + timeStamp;
    String key = consumerSecret + '&' + tokenSecret;
    String signature;
    if (signatureAlgorithm.equals(SignatureAlgorithm._HMAC_SHA256))
    {
        signature = computeShaHash(baseString, key, "HmacSHA256");
    } else {
        signature = computeShaHash(baseString, key, "HmacSHA1");
    }
    return signature;
}

3. Add logic for creating a TokenPassport SOAP header.

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

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

```java
public void testGetServerTimeWithTBA() throws Exception
{
    // Constants
    String sCompId = "1234567";
    String consumerKey = "71cc02b731f05895561ef0862d71553a3ac99498a947c3b7beaf4a1e4a29f7c4";
    String consumerSecret = "7278da58caf07f5c33601a60120310a58e948ef2a280f061825f85e1f26b";
    String tokenId = "89e08d9767c5ac85b3741572556705b54efc7960bad2478094a52f741b20d82";
    String tokenSecret = "060cd9ab3ffbe1e3d3918e90165f9f37ab12acc76b4691046e2d297d767ac2";
    String algorithm = SignatureAlgorithm._HMAC_SHA256;

    // Nonce and timestamp must be unique for each request. Also, make sure that the nonce value you
    // generate does not contain special characters.
    String nonce = RandomStringUtils.randomAlphanumeric(20);
    Long timeStamp = System.currentTimeMillis() / 1000L;

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

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

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

```xml
<ns:tokenPassport soap:actor="http://schemas.xmlsoap.org/soap/actor/next" soap:mustUnderstand="0"
xmlns:ns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <ns:account>1234567</ns:account>
  <ns:consumerKey>71cc02b731f05895561ef0862d71553a3ac99498a947c3b7beaf4a1e4a29f7c4</ns:consumerKey>
  <ns:token>89e08d9767c5ac85b374415725567d05b54ecf0960ad2470894a52f741028d2</ns:token>
  <ns:nonce>6obWq@tmYByVo6ddEKA1</ns:nonce>
  <ns:timestamp>1439829974</ns:timestamp>
  <ns:signature algorithm="HMAC_SHA256">fzGxUBu6SZvGqv5hk8P4ou2DPthSxlXJ4zJ1eCBOQKSA+</ns:signature>
</ns:tokenPassport>
```

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

---

## SOAP Web Services Governance for Token-Based Authentication

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

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

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

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

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

For detailed information about the changes in web services concurrency governance, see Web Services and RESTlet Concurrency Governance.
Token-Based Authentication Errors in SOAP Web Services

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

Ambiguous Authentication Errors

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

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

The error occurs in the following cases:

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

Other Invalid Login Errors

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

Issues with Nonce and Invalid Timestamps

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

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

Issues with Sending Multiple TokenPassports in a Single Request

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

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

Issues with Sending the Same TokenPassport in Multiple Requests

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

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

Troubleshooting Issues with Token-Based Authentication

See the following section for information about investigating authentication issues. For additional information, also see Using the Login Audit Trail.
Logging SOAP Requests and Responses

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

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

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

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

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

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

Blocking SOAP Web Services Requests

You can use an integration record to block SOAP web services requests that come from the corresponding application. However, when you use this method, you block only those requests that include data that identifies that integration record. For example, if you block Integration Record A, in essence you are blocking only those requests that include either Integration Record A's application ID or its token-based authentication details.
An application that is using the 2015.1 or an earlier endpoint is not required to send this data in requests. Requests that omit this data are handled by the Default Web Services Integrations record, which can also be blocked.

**To block web services requests:**

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

**SOAP Web Services Execution Log**

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

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

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

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

<table>
<thead>
<tr>
<th>Time</th>
<th>Execution Status</th>
<th>Action</th>
<th>API Version</th>
<th>Status</th>
<th>User Email</th>
<th>Records Processed</th>
<th>Records Published</th>
<th>Records Rejected by Query</th>
<th>Request</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>6/5/2013 11:13 AM</td>
<td>FAILED</td>
<td>add</td>
<td>2015.1</td>
<td>FINISHED</td>
<td><a href="mailto:nbaul@netsuite.com">nbaul@netsuite.com</a></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>view</td>
<td>view</td>
</tr>
<tr>
<td>6/5/2013 11:13 AM</td>
<td>FAILED</td>
<td>add</td>
<td>2017.2</td>
<td>FINISHED</td>
<td><a href="mailto:nbaul@netsuite.com">nbaul@netsuite.com</a></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>view</td>
<td>view</td>
</tr>
</tbody>
</table>

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

Another difference between the two logs is that the SOAP Web Services Usage Log is not specific to any one integration record. For more details about the SOAP Web Services Usage Log, see **Using the SOAP Web Services Usage Log**.
Note: In production environments, the data is accessible for 21 days. In sandbox environments, the data is accessible for seven days.

Distributing Integration Records

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

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

- Distributing by Auto-Installation
- Distributing by Bundling

Distributing by Auto-Installation

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

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

To distribute an integration record by using auto-installation:

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

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

See also Auto-Installation of Integration Records.

Distributing by Bundling

The integration record is supported for bundling with SuiteBundler.

This method may also be preferable if you have additional customizations that you want to distribute.
To include integration records in a bundle, select the appropriate records on the SuiteBundler’s Select Objects page. In the Object Types column, select Integrations > Integrations. Then select the appropriate records in the Choose Objects column. This column shows both active and inactive integration records.

See also Bundle Installation of Integration Records.

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

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

**Default Web Services Integrations Record**

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

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

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

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

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

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

**Note:** For information about integration records in general, see Integration Record Overview.

### Using Integration Records in Sandbox Accounts

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

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

**Important:** If you create an integration record in a sandbox account and then install it in a production account, you will not be the owner of the integration record in the production account. This means that you cannot update every field of the integration record in the production account. To have full control of the integration record in a production account, create the integration record directly in the production account.

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

**Recommended Development Process for Integration Records**

1. Create an integration record in a sandbox account and test your integration.
2. After testing, create a new integration record in your production account, and use the settings that you tested in your sandbox account.

When you update a specific integration record in another account, the behavior of the integration records in different accounts can be different. For more details, see the following sections:

- Common Fields Between Production and Sandbox Accounts
- Sharing Integration Records Between Sandbox and Production Accounts
- Sandbox Refresh Impact on Integration Records

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

### Common Fields Between Production and Sandbox Accounts

Integration records have fields that are specific for each account, and fields that are common between all accounts where the integration record is installed. The following fields are common:

- Application ID
Using Integration Records in Sandbox Accounts

- Name
- Description
- Authentication
- Consumer Key
- Consumer Secret

The Consumer Key and Consumer Secret are not directly editable. You cannot change them directly, but you can regenerate them if needed. The Consumer Key and Consumer Secret can only be regenerated in the account where the integration record was created.

⚠️ **Important:** The Application ID is not editable.

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

Sharing Integration Records Between Sandbox and Production Accounts

You can create an integration record in your sandbox account and install or auto-install it in your production account, or the other way around. In this case, you cannot fully manage the integration record in the account where you installed or auto-installed it, and you can only edit the State and the Note fields.

Sandbox Refresh Impact on Integration Records

When a sandbox account is refreshed, production data is copied to the sandbox account, and the earlier sandbox content is overwritten. A sandbox refresh is one of the few ways that an integration record can be uninstalled from a NetSuite account. That is, if a record was created in your sandbox account but not installed in your production account, it is removed as part of the sandbox refresh. After the refresh, the integration record can be auto-installed back into the sandbox account, if you have the application ID. However, you will not have full control of that integration record in the sandbox.

After every sandbox refresh, review the list of integration records, and block those you do not plan to use, because after the refresh all production integration records are copied to your sandbox accounts, including the production settings.

If you auto-install the integration record, be aware that data specific to the first installation is not retained. Data contained in the SOAP Web Services Execution Log, the State field, and the Note field is deleted. For details, see Auto-Installation of Integration Records.

Token-based Authentication Credentials and Accounts

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

- The Token Key and Token Secret used for TBA are always different in each account.
- The Token Key and Token Secret are deleted in your sandbox account after each sandbox refresh, and you must regenerate them. This has no impact on the Token Key and Token Secret used in your production account.

The Consumer Key and Consumer Secret can be shared between your sandbox and production accounts if you create an integration record in one account and install it on the other one. In this case, if you regenerate the Consumer Key or Consumer Secret in your sandbox account, they will be automatically changed in your production account and this can break your production account integrations.

Recommended practices:

- Create a separate integration record in each account to avoid changes in the production account when credentials are regenerated in a sandbox account.
- If you decide after testing that you need to make changes to an integration record, create a new record in your production account with those changes.

Not recommended practice:

- Do not create new integration records if you only need to change the settings of an existing integration record or only need to regenerate authentication credentials. Instead, make changes on an existing integration record in your sandbox account and then install the integration record in your production account.

Using Integration Records in Conjunction With SSO Calls

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

- **Inbound** single sign-on requests are treated exactly as requests that authenticate through user credentials. For example:
  - If you are using the 2015.2 or a later WSDL, inbound single sign-on requests must include an application ID. To permit an application to authenticate by using inbound SSO, check the User Credentials box on the appropriate integration record. You can also use the integration record to block requests that include the corresponding application ID.
  - If your requests use the 2015.1 WSDL or earlier, you can optionally include an application ID. Requests that omit application IDs are tracked as part of the Default Web Services Integrations record. (You can use the Default integration record to view an execution log of these requests. Additionally, if you block the Default record, these requests are rejected.)

- With **outbound** single sign-on requests (SuiteSignOn requests), you do not have to send an application ID, regardless of which endpoint you are using.

**Important:** As of 2018.2, new Inbound Single Sign-on (SSO) solutions are no longer enabled for use with SOAP web services for new customers or new partners. Current partners may continue to sell their existing Inbound SSO solutions to new customers. However, it is strongly recommended that you consider other options, and use token-based authentication (TBA) if possible. For more information, see Token-Based Authentication Details.

Removing Integration Records

When an integration record is created in a NetSuite account by a user filling out the Integration form, two things happen: A record is added to, or installed, in that NetSuite account. Additionally, within the broader NetSuite database, a definition of that record is created. This definition can never be deleted. However, in
Removing Integration Records

In some cases, the record can be uninstalled from the account in which it was created or the one to which it was distributed.

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

For details, see the following topics:
- Uninstalling an Integration Record from a NetSuite Account
- Removing Installed Integration Records from View

Uninstalling an Integration Record from a NetSuite Account

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

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

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

Removing Installed Integration Records from View

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

To remove an integration record from view:

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

Tracking Changes to Integration Records

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

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

For more details, see the following topics:
- Finding System Notes for an Integration Record
Special Cases Related to System Notes Logging

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

Finding System Notes for an Integration Record

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

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

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

Special Cases Related to System Notes Logging

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

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

Logging for the Last State Change Field

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

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

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

**Logging for the State Field**

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

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

Logging for the Reset Credentials Button

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

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

Generation of an Application ID Is Not logged

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

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

Logging for Records Shared by Sandbox and Production Accounts

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

**Notes Are Logged Only for Fields You Have Permission to Modify**

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

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

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

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

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

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

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

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

Role and Permission Considerations When Developing in SOAP Web Services

Due to SOAP web services’ reliance on the NetSuite role-based permissions, it is important for SOAP web services developers to put that into considerations during the design phase to ensure smooth deployments.

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

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

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

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

## Important:

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

### Assigning the SOAP Web Services Permission to a Role

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

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

**To assign the SOAP Web Services permission to a role:**

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

### Note:

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

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

Note also:

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

Setting a Default Role for a SOAP Web Services User

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

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

Only one default SOAP web services role can be assigned per user and only roles that contain the SOAP web services permission can be specified as a default SOAP web services role. Note that the user may be assigned a different role than those specified in their employee record. In other words, a user may have greater or lesser permissions using SOAP web services as compared to the UI.

- If neither the request nor the SOAP web services default role is set, then one of these roles is used, selected in the following order:
  - A non-Customer Center role in the account, indicated as the default role.
  - A non-Customer Center role in the account, selected based on the last login date.
  - A Customer Center role in the account, indicated as the default role.
  - A Customer Center role in the account, selected based on the last login date.

**Note:** All standard roles have the SOAP web services permission by default when the SOAP web services feature is enabled. Custom roles, however, must be explicitly set to have SOAP web services permissions. For information about creating custom roles, see the help topic Customizing or Creating NetSuite Roles.

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

1. Go to Setup > Integration > Integration Management > SOAP Web Services Preferences.
2. Select the user from the Name list.
3. Select the default role to use for SOAP web services requests for this user.
   - The internal ID for the selected role automatically populates the ID field.
4. Click Done.
5. Click Save.

Setting a Web Services Only Role for a User

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

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

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

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

To designate a role as Web Services Only:

Note: A role designated as Web Services Only prohibits the user from accessing NetSuite through the UI. The intention is to enhance the security of your integrations. However, checking the Web Services Only box does not exclude access to NetSuite by other non-UI methods. Be aware that a Web Services Only role could combine with other access permissions. For example, if you designate a role as Web Services Only, and that role (or a user) also has SuiteAnalytics Connect permissions, the user could also access NetSuite through ODBC.

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

When to Set the Web Services Only Role

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

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

Customer Center, Vendor Center, and Partner Center Roles

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

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

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

**To remove SOAP web services permissions:**

1. Go to Setup > Users/Roles > Manage Roles.
2. Locate the role you want to modify.
3. Click the corresponding **Edit** or **Customize** link.
4. Click the **Setup** subtab.
5. In the **Permissions** dropdown list, click the **Web Services** line to make it editable. The page updates to include several buttons below the line.
6. Click the **Remove** button.
7. Click **Save**.

**Internal IDs Associated with Roles**

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

**Note:** The administrator role is not displayed in the Manage Roles page. The internal ID for the administrator role is 3 in every NetSuite account.

For information about the permissions associated with standard roles, see the help topic Standard Roles Permissions Table.

You can use the internal IDs of roles in the Passport object, which is used for authentication. For information about the available authentication methods, see Authentication for SOAP Web Services.

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

**Important:** Administrators and users with Full Access roles must use two-factor authentication in newly provisioned accounts. For more information, see the help topic 2FA in the NetSuite Application. Using the Administrator role for web services is not recommended. Using the Full Access role at all is not recommended. As of 2019.1, changes have been made to the Full Access role, in preparation for its deprecation. For details, see the help topic Plan to Deprecate the Full Access Role in the 2019.1 Release Notes.
Records, Fields, Forms, and Sublists in SOAP Web Services

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

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

Records in SOAP Web Services

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

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

NetSuite records are divided into the following broad categories:

- Record Types
- Search Records
- Subrecords

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

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

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

Record Types

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

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Corresponds to the SearchStringField type</td>
</tr>
<tr>
<td>Int</td>
<td>Corresponds to the SearchTextNumberField type</td>
</tr>
<tr>
<td>Double</td>
<td>Corresponds to the SearchDoubleField type</td>
</tr>
<tr>
<td>Boolean</td>
<td>Corresponds to the SearchBooleanField type</td>
</tr>
<tr>
<td>Datetime</td>
<td>Corresponds to the SearchDateField type</td>
</tr>
<tr>
<td>MultiSelectRef</td>
<td>Corresponds to the SearchMultiSelectField type</td>
</tr>
<tr>
<td>MultiSelectEnum</td>
<td>Corresponds to the SearchEnumMultiSelectField type</td>
</tr>
</tbody>
</table>

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

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

Subrecords

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

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

The following preferences apply to subrecords:

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

Using Internal IDs, External IDs, and References

Each record in NetSuite can be uniquely identified by its record type in combination with either an external ID or a system-generated NetSuite internal ID.
Important: To see the internal IDs for all fields, records, lists, and custom forms, enable the Show Internal IDs preference. See Enabling the Show Internal IDs Preference. External IDs are not displayed on records by default. However, you can search for external IDs, or you can add a custom field to the record types where you want to the external ID to be displayed in the UI. For information about searching in SOAP web services, see Basic Searches in SOAP Web Services. For information about working with custom fields, see the help topic Custom Fields.

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

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

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

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

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

Uniqueness of Internal and External IDs

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

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

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

Internal IDs Do Not Indicate When Record Was Created

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

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

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

**Internal IDs for NetSuite Record Types**

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

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

<table>
<thead>
<tr>
<th>Record Type</th>
<th>typeId</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNT</td>
<td>-112</td>
</tr>
<tr>
<td>ACCOUNTINGPERIOD</td>
<td>-105</td>
</tr>
<tr>
<td>ADDRESS</td>
<td>-289</td>
</tr>
<tr>
<td>ADVINTERCOMPANYJOURNALENTRY</td>
<td>-30</td>
</tr>
<tr>
<td>APPDEFINITION</td>
<td>-255</td>
</tr>
<tr>
<td>APPPACKAGE</td>
<td>-254</td>
</tr>
<tr>
<td>ASSEMBLYBUILD</td>
<td>-30</td>
</tr>
<tr>
<td>ASSEMBLYITEM</td>
<td>-10</td>
</tr>
<tr>
<td>ASSEMBLYUNBUILD</td>
<td>-30</td>
</tr>
<tr>
<td>BILLINGACCOUNT</td>
<td>-333</td>
</tr>
<tr>
<td>BILLINGSCHEDULE</td>
<td>-141</td>
</tr>
<tr>
<td>BIN</td>
<td>-242</td>
</tr>
<tr>
<td>BINTRANSFER</td>
<td>-30</td>
</tr>
<tr>
<td>BINWORKSHEET</td>
<td>-30</td>
</tr>
<tr>
<td>BOM</td>
<td>-422</td>
</tr>
<tr>
<td>BOMREVISION</td>
<td>-423</td>
</tr>
<tr>
<td>BUDGETCATEGORY</td>
<td>-396</td>
</tr>
<tr>
<td>BUDGETIMPORT</td>
<td>-420</td>
</tr>
<tr>
<td>CALENDAREVENT</td>
<td>-20</td>
</tr>
<tr>
<td>CAMPAIGN</td>
<td>-24</td>
</tr>
<tr>
<td>CAMPAIGNAUDIENCE</td>
<td>-142</td>
</tr>
<tr>
<td>CAMPAIGNCATEGORY</td>
<td>-143</td>
</tr>
<tr>
<td>Record Type</td>
<td>typeId</td>
</tr>
<tr>
<td>----------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>CAMPAIGNCHANNEL</td>
<td>-144</td>
</tr>
<tr>
<td>CAMPAIGNFAMILY</td>
<td>-145</td>
</tr>
<tr>
<td>CAMPAIGNOFFER</td>
<td>-146</td>
</tr>
<tr>
<td>CAMPAIGNRESPONSE</td>
<td>-130</td>
</tr>
<tr>
<td>CAMPAIGNSEARCHENGINE</td>
<td>-148</td>
</tr>
<tr>
<td>CAMPAIGNSUBSCRIPTION</td>
<td>-149</td>
</tr>
<tr>
<td>CAMPAIGNVERTICAL</td>
<td>-150</td>
</tr>
<tr>
<td>CASHREFUND</td>
<td>-30</td>
</tr>
<tr>
<td>CASHSALE</td>
<td>-30</td>
</tr>
<tr>
<td>CHARGE</td>
<td>-290</td>
</tr>
<tr>
<td>CHECK</td>
<td>-30</td>
</tr>
<tr>
<td>CLASSIFICATION</td>
<td>-101</td>
</tr>
<tr>
<td>CONSOLIDATEDEXCHANGERATE</td>
<td>-322</td>
</tr>
<tr>
<td>CONTACT</td>
<td>-6</td>
</tr>
<tr>
<td>CONTACTCATEGORY</td>
<td>-158</td>
</tr>
<tr>
<td>CONTACTROLE</td>
<td>-157</td>
</tr>
<tr>
<td>COSTCATEGORY</td>
<td>-155</td>
</tr>
<tr>
<td>CREDITMEMO</td>
<td>-30</td>
</tr>
<tr>
<td>CRMCUSTOMFIELD</td>
<td>-124</td>
</tr>
<tr>
<td>CURRENCY</td>
<td>-122</td>
</tr>
<tr>
<td>CUSTOMER</td>
<td>-2</td>
</tr>
<tr>
<td>CUSTOMERCATEGORY</td>
<td>-109</td>
</tr>
<tr>
<td>CUSTOMERDEPOSIT</td>
<td>-30</td>
</tr>
<tr>
<td>CUSTOMERMESSAGE</td>
<td>-161</td>
</tr>
<tr>
<td>CUSTOMERPAYMENT</td>
<td>-30</td>
</tr>
<tr>
<td>CUSTOMERREFUND</td>
<td>-30</td>
</tr>
<tr>
<td>CUSTOMERSTATUS</td>
<td>-104</td>
</tr>
<tr>
<td>CUSTOMLIST</td>
<td>-123</td>
</tr>
<tr>
<td>CUSTOMRECORDCUSTOMFIELD</td>
<td>-124</td>
</tr>
<tr>
<td>CUSTOMRECORDDTYPE</td>
<td>-123</td>
</tr>
<tr>
<td>CUSTOMTRANSACTION</td>
<td>-30</td>
</tr>
<tr>
<td>CUSTOMTRANSACTION</td>
<td>-30</td>
</tr>
<tr>
<td>Record Type</td>
<td>typeId</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>CUSTOMTRANSACTION</td>
<td>-30</td>
</tr>
<tr>
<td>CUSTOMTRANSACTION</td>
<td>-30</td>
</tr>
<tr>
<td>DEPARTMENT</td>
<td>-102</td>
</tr>
<tr>
<td>DEPOSIT</td>
<td>-30</td>
</tr>
<tr>
<td>DEPOSITAPPLICATION</td>
<td>-30</td>
</tr>
<tr>
<td>DESCRIPTIONITEM</td>
<td>-10</td>
</tr>
<tr>
<td>DISCOUNTITEM</td>
<td>-10</td>
</tr>
<tr>
<td>DOWNLOADITEM</td>
<td>-10</td>
</tr>
<tr>
<td>EMPLOYEE</td>
<td>-4</td>
</tr>
<tr>
<td>ENTITYCUSTOMFIELD</td>
<td>-124</td>
</tr>
<tr>
<td>ESTIMATE</td>
<td>-30</td>
</tr>
<tr>
<td>EXPENSECATEGORY</td>
<td>-126</td>
</tr>
<tr>
<td>EXPENSEREPOR</td>
<td>-30</td>
</tr>
<tr>
<td>FAIRVALUEPRICE</td>
<td>-332</td>
</tr>
<tr>
<td>GENERALTOKEN</td>
<td>-540</td>
</tr>
<tr>
<td>GIFTCERTIFICATEITEM</td>
<td>-10</td>
</tr>
<tr>
<td>GLOBALACCOUNTMAPPING</td>
<td>-250</td>
</tr>
<tr>
<td>HCMJOB</td>
<td>-355</td>
</tr>
<tr>
<td>INBOUNDSHIPMENT</td>
<td>-427</td>
</tr>
<tr>
<td>INTERCOMPANYJOURNALENTRY</td>
<td>-30</td>
</tr>
<tr>
<td>INTERCOMPANYTRANSFERORDER</td>
<td>-30</td>
</tr>
<tr>
<td>INVENTORYADJUSTMENT</td>
<td>-30</td>
</tr>
<tr>
<td>INVENTORYCOSTREVALUATION</td>
<td>-30</td>
</tr>
<tr>
<td>INVENTORYDETAIL</td>
<td>-260</td>
</tr>
<tr>
<td>INVENTORYITEM</td>
<td>-10</td>
</tr>
<tr>
<td>INVENTORYNUMBER</td>
<td>-266</td>
</tr>
<tr>
<td>INVENTORYTRANSFER</td>
<td>-30</td>
</tr>
<tr>
<td>INVOICE</td>
<td>-30</td>
</tr>
<tr>
<td>ISSUE</td>
<td>-26</td>
</tr>
<tr>
<td>ITEM</td>
<td>-10</td>
</tr>
<tr>
<td>ITEMACCOUNTMAPPING</td>
<td>-251</td>
</tr>
<tr>
<td>ITEMCUSTOMFIELD</td>
<td>-124</td>
</tr>
<tr>
<td>Record Type</td>
<td>typeId</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>ITEMDEMANDPLAN</td>
<td>-246</td>
</tr>
<tr>
<td>ITEMFULFILLMENT</td>
<td>-30</td>
</tr>
<tr>
<td>ITEMGROUP</td>
<td>-10</td>
</tr>
<tr>
<td>ITEMNUMBERCUSTOMFIELD</td>
<td>-124</td>
</tr>
<tr>
<td>ITEMOPTIONCUSTOMFIELD</td>
<td>-124</td>
</tr>
<tr>
<td>ITEMRECEIPT</td>
<td>-30</td>
</tr>
<tr>
<td>ITEMREVISION</td>
<td>-269</td>
</tr>
<tr>
<td>ITEMSUPPLYPLAN</td>
<td>-247</td>
</tr>
<tr>
<td>JOB</td>
<td>-7</td>
</tr>
<tr>
<td>JOBTYPE</td>
<td>-177</td>
</tr>
<tr>
<td>JOURNALENTRY</td>
<td>-30</td>
</tr>
<tr>
<td>KITITEM</td>
<td>-10</td>
</tr>
<tr>
<td>LOCATION</td>
<td>-103</td>
</tr>
<tr>
<td>LOTNUMBEREDASSEMBLYITEM</td>
<td>-10</td>
</tr>
<tr>
<td>LOTNUMBEREDINVENTORYITEM</td>
<td>-10</td>
</tr>
<tr>
<td>MANUFACTURINGCOSTTEMPLATE</td>
<td>-294</td>
</tr>
<tr>
<td>MANUFACTURINGOPERATIONTASK</td>
<td>-36</td>
</tr>
<tr>
<td>MANUFACTURINGROUTING</td>
<td>-288</td>
</tr>
<tr>
<td>MARKUPITEM</td>
<td>-10</td>
</tr>
<tr>
<td>Nexus</td>
<td>-400</td>
</tr>
<tr>
<td>NONINVENTORYPURCHASEITEM</td>
<td>-10</td>
</tr>
<tr>
<td>NONINVENTORYRESALEITEM</td>
<td>-10</td>
</tr>
<tr>
<td>NONINVENTORIESALEITEM</td>
<td>-10</td>
</tr>
<tr>
<td>Note</td>
<td>-303</td>
</tr>
<tr>
<td>Notetype</td>
<td>-180</td>
</tr>
<tr>
<td>Opportunity</td>
<td>-31</td>
</tr>
<tr>
<td>OTHERCHARGEPURCHASEITEM</td>
<td>-10</td>
</tr>
<tr>
<td>OTHERCHARGERESALEITEM</td>
<td>-10</td>
</tr>
<tr>
<td>OTHERCHARGESALEITEM</td>
<td>-10</td>
</tr>
<tr>
<td>OTHERCUSTOMFIELD</td>
<td>-124</td>
</tr>
<tr>
<td>OTHERNAMECATEGORY</td>
<td>-181</td>
</tr>
<tr>
<td>Partner</td>
<td>-5</td>
</tr>
<tr>
<td>Record Type</td>
<td>typeId</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------</td>
</tr>
<tr>
<td>PARTNERCATEGORY</td>
<td>-182</td>
</tr>
<tr>
<td>PAYCHECK</td>
<td>-30</td>
</tr>
<tr>
<td>PAYCHECKJOURNAL</td>
<td>-30</td>
</tr>
<tr>
<td>PAYMENTCARD</td>
<td>-538</td>
</tr>
<tr>
<td>PAYMENTCARDTOKEN</td>
<td>-539</td>
</tr>
<tr>
<td>PAYMENTITEM</td>
<td>-10</td>
</tr>
<tr>
<td>PAYMENTMETHOD</td>
<td>-183</td>
</tr>
<tr>
<td>PAYROLLITEM</td>
<td>-265</td>
</tr>
<tr>
<td>PHONECALL</td>
<td>-22</td>
</tr>
<tr>
<td>PRICELEVEL</td>
<td>-186</td>
</tr>
<tr>
<td>PRICINGGROUP</td>
<td>-187</td>
</tr>
<tr>
<td>PROJECTTASK</td>
<td>-27</td>
</tr>
<tr>
<td>PROMOTIONCODE</td>
<td>-121</td>
</tr>
<tr>
<td>PURCHASEORDER</td>
<td>-30</td>
</tr>
<tr>
<td>PURCHASEREQUISITION</td>
<td>-30</td>
</tr>
<tr>
<td>RESOURCEALLOCATION</td>
<td>-28</td>
</tr>
<tr>
<td>RETURNAUTHORIZATION</td>
<td>-30</td>
</tr>
<tr>
<td>SALESORDER</td>
<td>-30</td>
</tr>
<tr>
<td>SALESROLE</td>
<td>-191</td>
</tr>
<tr>
<td>SALESTAXITEM</td>
<td>-128</td>
</tr>
<tr>
<td>SERIALIZEDASSEMBLYITEM</td>
<td>-10</td>
</tr>
<tr>
<td>SERIALIZEDINVENTORYITEM</td>
<td>-10</td>
</tr>
<tr>
<td>SERVICEPURCHASEITEM</td>
<td>-10</td>
</tr>
<tr>
<td>SERVICESALEITEM</td>
<td>-10</td>
</tr>
<tr>
<td>SERVICESALEITEM</td>
<td>-10</td>
</tr>
<tr>
<td>SOLUTION</td>
<td>-25</td>
</tr>
<tr>
<td>STATISTICALJOURNALENTRY</td>
<td>-30</td>
</tr>
<tr>
<td>SUBSIDIARY</td>
<td>-117</td>
</tr>
<tr>
<td>SUBTOTALITEM</td>
<td>-10</td>
</tr>
<tr>
<td>SUPPORTCASE</td>
<td>-23</td>
</tr>
<tr>
<td>SUPPORTCASEISSUE</td>
<td>-151</td>
</tr>
<tr>
<td>SUPPORTCASEORIGIN</td>
<td>-152</td>
</tr>
</tbody>
</table>
### External IDs Overview

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

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

- **Maintaining client ID relationships**

<table>
<thead>
<tr>
<th>Record Type</th>
<th>typeId</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUPPORTCASEPRIORITY</td>
<td>-153</td>
</tr>
<tr>
<td>SUPPORTCASESTATUS</td>
<td>-132</td>
</tr>
<tr>
<td>SUPPORTCASETYPE</td>
<td>-154</td>
</tr>
<tr>
<td>TASK</td>
<td>-21</td>
</tr>
<tr>
<td>TERM</td>
<td>-199</td>
</tr>
<tr>
<td>TIMEBILL</td>
<td>-256</td>
</tr>
<tr>
<td>TIMEENTRY</td>
<td>-295</td>
</tr>
<tr>
<td>TIMESHEET</td>
<td>-292</td>
</tr>
<tr>
<td>TRANSACTION</td>
<td>-30</td>
</tr>
<tr>
<td>TRANSACTIONBODYCUSTOMFIELD</td>
<td>-124</td>
</tr>
<tr>
<td>TRANSACTIONCOLUMNCUSTOMFIELD</td>
<td>-124</td>
</tr>
<tr>
<td>TRANSFERORDER</td>
<td>-30</td>
</tr>
<tr>
<td>UNITSTYPE</td>
<td>-201</td>
</tr>
<tr>
<td>USAGE</td>
<td>-362</td>
</tr>
<tr>
<td>VENDOR</td>
<td>-3</td>
</tr>
<tr>
<td>VENDORBILL</td>
<td>-30</td>
</tr>
<tr>
<td>VENDORCATEGORY</td>
<td>-110</td>
</tr>
<tr>
<td>VENDORCREDIT</td>
<td>-30</td>
</tr>
<tr>
<td>VENDORPAYMENT</td>
<td>-30</td>
</tr>
<tr>
<td>VENDORRETURNAUTHORIZATION</td>
<td>-30</td>
</tr>
<tr>
<td>WINLOSSREASON</td>
<td>-203</td>
</tr>
<tr>
<td>WORKORDER</td>
<td>-30</td>
</tr>
<tr>
<td>WORKORDERCLOSE</td>
<td>-30</td>
</tr>
<tr>
<td>WORKORDERCOMPLETION</td>
<td>-30</td>
</tr>
<tr>
<td>WORKORDERISSUE</td>
<td>-30</td>
</tr>
</tbody>
</table>
In cases where a client application already maintains references between records, set the externalId attribute for each record during imports. In subsequent API calls, you can then reference associated records by the known external ID.

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

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

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

| Important: | External IDs are not displayed on records by default. However, you can search for external IDs, or you can add a custom field to the record types where you want the external ID to be displayed in the UI. For information about searching in SOAP web services, see Basic Searches in SOAP Web Services. For information about working with custom fields, see the help topic Custom Fields. |

The record types that do not support external ID are:

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

**Guidelines for External IDs**

Consider the following guidelines for external IDs:

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

Although records of a particular type may be used in multiple integration scenarios, each record instance can only have a single external ID value. To maintain data integrity, only a single integrated application can set and update external ID values for each record type. External ID values for all records of a particular type must all come from the same external application.

External IDs are case insensitive in web services references. For details, see Using Internal IDs, External IDs, and References.

Shared Internal and External IDs

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

Note: For information about displaying internal IDs, see Setting the Internal ID Preference.

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

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

<table>
<thead>
<tr>
<th>Record Group</th>
<th>Record Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
<td>Account; Tax Account</td>
</tr>
<tr>
<td>Allocation</td>
<td>Allocation Schedule; Intercompany Allocation Schedule</td>
</tr>
<tr>
<td>Schedule</td>
<td></td>
</tr>
<tr>
<td>Entity</td>
<td>Contact; Customer; Employee; Entity; Entity Group; Job; Lead; Other Name; Partner; Project Template; Prospect; Vendor</td>
</tr>
<tr>
<td>Entity Status</td>
<td>Customer Status; Job Status</td>
</tr>
<tr>
<td>Event</td>
<td>Activity; Campaign; Case; Event; Issue; Manufacturing Operation Task; Phone Call; Project Task; Resource Allocation; Sales Campaign; Solution; Task</td>
</tr>
<tr>
<td>Inbound</td>
<td>Bulk Ownership Transfer; Inbound Shipment; Receive Inbound Shipment</td>
</tr>
<tr>
<td>Shipment</td>
<td></td>
</tr>
<tr>
<td>Inventory Item</td>
<td>Assembly/Bill of Materials; Description Item; Discount Item; Download Item; Expense Item; Gift Certificate Item; Inventory Part; Item; Item Group; Kit Item; Lot Numbered Assembly/Bill of Materials; Lot Numbered Inventory Item; Markup Item; Non-Inventory Part; Non-inventory Item for Purchase; Non-inventory Item for Resale; Non-inventory Item for Sale; Other Charge Item for Purchase; Other Charge Item for Resale; Other Charge Item for Sale; Payment Item; Serialized Assembly/Bill of Materials; Serialized Inventory Item; Service; Service Item for Purchase; Service Item for Resale; Service Item for Sale; Subscription Plan; Subtotal Item; Tax Group</td>
</tr>
<tr>
<td>Payroll Batch</td>
<td>Payroll Batch; Payroll Batch - Add Employees</td>
</tr>
<tr>
<td>Project Revenue</td>
<td>Fixed Amount Project Revenue Rule; Labor Based Project Revenue Rule; Percent Complete Project Revenue Rule</td>
</tr>
</tbody>
</table>
### Fields in SOAP Web Services

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Corresponds to the xsd:string type in the XML Schema</td>
</tr>
<tr>
<td>Int</td>
<td>Corresponds to the xsd:int type in the XML Schema</td>
</tr>
<tr>
<td>Double</td>
<td>Corresponds to the xsd:double in the XML Schema</td>
</tr>
<tr>
<td>Boolean</td>
<td>Corresponds to the xsd:boolean type in the XML Schema and has valid values of true or false. If not explicitly set to either true or false, then set as false.</td>
</tr>
<tr>
<td>Datetime</td>
<td>Corresponds to the xsd:dateTime type in the XML Schema which conforms to the ISO 8601 standard.</td>
</tr>
<tr>
<td>RecordRef</td>
<td>Corresponds to the RecordRef type in the XML Schema. References an nsKey value for any other record in the system including system defined constants that are controlled by the system.</td>
</tr>
<tr>
<td>Enum</td>
<td>Corresponds to a specific type defined as an enum in the XSD that represents system constants that are also available in the UI.</td>
</tr>
<tr>
<td>WsEnum</td>
<td>Corresponds to a specific type defined as an enum in the XSD that represents system constants that are NOT available in the UI.</td>
</tr>
<tr>
<td>List</td>
<td>A List references a type that is a list and should be explicitly defined in the in the XML Schema as a type. A list can either be null (if it is an optional field), or it must contain at least one entry unless otherwise noted.</td>
</tr>
</tbody>
</table>

Custom fields are those that have been created by NetSuite users to customize their accounts. Custom fields can be added to an account using SuiteBuilder point-and-click customization tools. They can also be added through the SOAP web services `add` operation. Custom fields must be one of the types defined in `Custom Field Types`. 
On records, custom fields are contained in the customFieldList property. For more details, see CustomFieldList.

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

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

- If a **standard** field is set as mandatory in a custom form, the "requiredness" of the field is honored in SOAP web services. If you do not provide value in your SOAP web services request for a field that is set as mandatory in a custom form, an error message is returned. For more information on NetSuite form types, see the help topic Custom Forms.

- Customizations made to **standard** fields are honored in SOAP web services. For example, if a standard field is set as disabled, it is not settable through SOAP web services, either. If you try to set such a field through SOAP web services, an error message is returned.

- If a **standard** field is set to NOT show in the UI, it is not settable through SOAP web services, either. If you try to set such a field through SOAP web services, an error message is returned.

- **Custom** display only (inline) and disabled fields are not settable through SOAP web services because these are NOT settable through the UI. If you provide a value in your SOAP web services request for such a field, an error message is returned. For information on field display types, see the help topic Setting Display Options for Custom Fields.

- The behavior of **custom** hidden fields varies depending on the endpoint being used. If you are using any endpoint since the 2011.2 endpoint (including 2011.2), the get operation does not return the value of custom hidden fields. If you are using an older endpoint, these values are gettable. By comparison, in the UI, these values cannot be seen.

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

  Custom hidden fields cannot be set, either in the user interface or using SOAP web services, regardless of the endpoint.

- Defaulted fields that are set to blank on a SOAP web services update will stay blank on update.

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

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

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

## Datetime Fields

You can work with datetime fields using the dateTime type in NetSuite. See the following sections for information about working with the dateTime type.

### Using the dateTime Type in NetSuite

When setting a date field using SOAP web services, you must enter a datetime value. The time and the time zone following the date are used to determine the resulting date. When getting the date field
through SOAP web services, your preferred time zone influences the resulting datetime. Midnight in the user's time zone of the stored date (which is also displayed in the UI) is converted to Pacific Standard Time (PST) or Pacific Daylight Time (PDT), and sent to the SOAP web services user.

**Note:** During winter months, the Pacific Time Zone is eight hours behind Coordinated Universal Time (UTC-8) and is referred to as Pacific Standard Time (PST). In summer months, the Pacific Time Zone is seven hours behind Coordinated Universal Time (UTC-7) and is referred to as Pacific Daylight Time (PDT).

The following table gives several examples.

<table>
<thead>
<tr>
<th>Datetime set in the date field</th>
<th>Time zone of the user</th>
<th>Date shown in the UI</th>
<th>Datetime when getting the record</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-02-21T00:00:00.000-06:00</td>
<td>GMT-06:00</td>
<td>2/21/2017</td>
<td>2017-02-20T22:00:00.000-08:00</td>
</tr>
<tr>
<td>2017-08-21T00:00:00.000-05:00</td>
<td>GMT-05:00 1</td>
<td>8/21/2017</td>
<td>2017-08-20T22:00:00.000-07:00</td>
</tr>
<tr>
<td>2017-02-21T00:00:00.000-05:00</td>
<td>GMT-08:00</td>
<td>2/20/2017 3</td>
<td>2017-02-20T00:00:00.000-08:00</td>
</tr>
<tr>
<td>2017-08-21T00:00:00.000-07:00</td>
<td>GMT-05:00 1</td>
<td>8/21/2017</td>
<td>2017-08-20T22:00:00.000-07:00</td>
</tr>
</tbody>
</table>

Notes for the example table:

1. The user has set their time zone to GMT-6, but daylight saving time (DST) affects the result.
2. This is midnight of the date from the UI that is converted to PST or PDT. Note the time difference, which equals the difference between the user's preferred time zone and PST or PDT.
3. The datetime which is to be saved is not greater than midnight in the user's preferred time zone.

**Guidelines for the dateTime Type**

Consider the following guidelines when using the dateTime type:

- To avoid problems with conversions, set the time zone of the SOAP web services employee to "(GMT-08:00) Pacific Time (US & Canada)" in Home > Set Preferences (Administrator). Also, submit the datetimes for date fields as midnight and in PST or PDT. For more information, see Setting Date and Time Format for SOAP Web Services. This way you can avoid conversions or truncations, and what is submitted will be identical to what is retrieved through SOAP web services.
- When changing time zone preferences the user must log in again using SOAP web services for the change to take effect. If the same session is used, the old time zone is used as well. Caching may also occur for a period after changing time zones.
- Users setting the field value need to pay attention to the DST offset, as it influences the decision point (midnight of user's time zone) according to which value is set to the field.
- If the time zone specified in a set field request is different from the time zone of the user setting the field, the datetime is first converted to the time zone of the user, and the date portion is stored. The time is then set to midnight and returned in PST or PDT. See the second row of the preceding example table for an example.
- When posting dateTime values to NetSuite, it is recommended that you indicate the time zone. If no time zone is indicated, the PST or PDT time zone is used. Note that NetSuite does not use GMT.

**Setting Date and Time Format for SOAP Web Services**

The following is an example of setting datetime, date, or time fields in SOAP web services, using Date-Time-Timezone:
If the user does not indicate the time zone in the request, the NetSuite system time zone is used:

(PST UTC-8/PDT UTC-07): `<xmp><s0:endDate>2013-10-30T20:01:24.405</s0:endDate></xmp>`

Users can indicate a UTC time zone by adding the 'Z' character at the end of the DateTime format:

`<xmp><s0:endDate>2013-10-30T20:01:24.405Z</s0:endDate></xmp>`

**Working with Daylight Saving Time in SOAP Web Services**

At specified dates in the calendar year, the time is adjusted for daylight saving in some time zones. This causes a shift from standard time (such as CST) to daylight saving time (such as CDT), and inversely, depending on the time of the year. These changes come with a gain (start of daylight saving time) or loss (start of standard time) of one hour in time difference.

The NetSuite time zone settings in Home > Set Preferences (Administrator) are based on standard time settings. You must keep in mind the required adjustments during daylight saving time in the year. For example, if your time zone setting in shows (GMT-06:00) Central Time (US & Canada), this is perceived as -05:00 in SOAP web services context during daylight saving time, but -06:00 in standard time.

Therefore, if you send a SOAP web services request indicating CST (-6) when daylight saving time is in effect, as in the following example: `<tranDate xsi:type="xsd:dateTime">2015-04-13T22:26:26.923-06:00</tranDate>`, the datetime field on the target record is set using CDT (-5) i.e.23:26:26.923 instead of the intended 22:26:26.923.

**Important:** If the time is close to midnight, the date may change to the next day after the conversion.

---

**CustomFieldList**

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

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

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

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

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

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

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

Sample .NET Code

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

Sample Java Code

```java
Customer cust = new Customer();
cust.setInternalId("373");
NullField nfl = new NullField(new String[]{"custentity9"});
```
Fields in SOAP Web Services

```
cust.setNullFieldList(nfl);
port.update(cust);
```

Custom Fields and Joined Searches

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

User Defined Lists

In SOAP web services, many fields require internal ID values that correspond to an item in a user-defined list. To locate the internal ID of a specific value in a user-defined list, you must have the **Show Internal IDs** preference enabled in your account. For details, see [Enabling the Show Internal IDs Preference](#).

After enabling the **Show Internal IDs** preference, you can then navigate to the appropriate list within NetSuite. The internal ID values (also referred to as nsKeys) are displayed for each list item.

For example, the Status (entityStatus) field on the customer record takes an internal ID value from the Customer Status list, which is a user-defined list located at Setup > Sales > Customer Statuses. If you navigate to this page in NetSuite, you can see the internal ID values for each item in the column called **Internal ID** (see figure).

![Customer Status List](image)

**Note:** Some user-defined lists can also be edited through SOAP web services to modify the values for the list. For details, see the help topic [Other Lists](#).

Enumerated Lists

If values being returned for WSEnum and Enum fields in a get or search operation do NOT match the values enumerated in the schema, the following warning is returned:
Fields in SOAP Web Services

In these cases, the existing data is corrupt and must be corrected before it can be retrieved using SOAP web services.

Field Lengths

Most string fields have a maximum character limit. If the limit is exceeded, an error is returned indicating which field exceeded the limit and the maximum number of characters allowed for that field.

Field Level Errors

If a required field is missing for a specific record within a request, an error is generated. The associated record is not processed and an appropriate error status code is returned.

Only the records without errors are processed. If multiple records are submitted within the same request, records without errors are processed; records with errors are not processed.

Required Fields

Required fields in the UI do not necessarily correspond to required fields in SOAP web services. This is because there can be standard defaults that are applied if the field is not populated. For example, in the CalendarEvent record type, the eventAccess field is required in the UI. However, it is optional in SOAP web services because it has a default value of Public.

Important: The Required column in the SOAP Schema Browser lists T or F (for true or false) to specify if a field is required by the form in the user interface. Refer to the Cardinality column to see whether the XSD imposes a requirement for the field. For information on using the SOAP Schema Browser, see the help topic SOAP Schema Browser.

Fields and Operations

For simplicity, the SOAP WSDL is designed with all available fields listed for each record in each corresponding XSD. There is no differentiation as to what field is available for each individual operation. For example, add operations take a separate set of field values than the corresponding update operation for some records. If your SOAP web services request includes a field value for a field that is unavailable, an error is thrown for that submission.

If you are using SOAP web services for data migration where there may be fields that need to be populated that are unavailable during an add operation, you should perform two consecutive requests. Submit an initial add or addList request, with values for all fields available for an add operation, followed by an update or updateList request, with values for the fields available only during an update operation.

Default Field Values

The system provides default values only for fields that are not required. When applying a default value, the system first tries to use a specified value. If none is given, the system uses the default. If no default is given, the system enters null.

User-defined defaults through the UI do not apply to SOAP web services.
Fields in SOAP Web Services

Hidden Fields

The SOAP web services API includes fields that are not visible in the NetSuite UI. In some cases, these fields have been hidden through SuiteBuilder point-and-click customization. In other cases, hidden fields are hidden by NetSuite. These types of hidden fields are used primarily for storing system-generated information such as dates.

For example, every SOAP-supported record contains either a `dateCreated` or `createdDate` field. These fields are not writeable. By default, these fields are populated with a system-generated timestamp. Note that system-generated dates and times are preserved for audit trail purposes.

**Note:** The Use Defaults Web service preference does not affect the behavior of hidden fields. Even if the Use Defaults preference is not enabled, hidden fields are populated when a value is not supplied in the request. Also, for audit purposes, the true system date is always recorded for each record and cannot be overridden.

**Important:** Calls to get or search (when the full record is returned) return the record in edit mode.

Forms in SOAP Web Services

If you are building a generic SOAP web services application, to ensure that your applications are account independent, NetSuite recommends that you create custom "Web-services-only" forms. Use custom forms for SOAP web services requests by specifying the form in the `customForm` element of a record. This will alleviate problems associated with customer-specific customizations to forms used in the UI that may break your SOAP web services application code.

**Note:** If you are unfamiliar with NetSuite custom forms, see the help topic Custom Forms. For information on creating custom forms that you can designate as "web-services-specific," see the help topic Creating Custom Entry and Transaction Forms.

For all record types that support custom forms, if a form is specified in the SOAP web services request, that form will be used for validation. Note that it is generally best practice to specify the custom form in all SOAP web services update operations. Also be aware that no record type is needed when specifying a custom form. You only need to specify the internal ID of the custom form.

To get the internal ID of a custom form, in the UI, go to Customization > Forms > [form type], where form type is either an entry or transaction form. The internal ID appears in the Internal ID column if the Show Internal IDs preference is enabled. (For steps on enabling this preference, see Enabling the Show Internal IDs Preference.)

To specify a form in SOAP web services, see the following C# snippet, which shows how to associate a particular form with a Customer record.

```csharp
// create a customer object
Customer customer = new Customer();

// create a custom form object
```
RecordRef customFormRef = new RecordRef();

// set the internal ID of the custom form. Get the internal ID from
// the UI
customFormRef.internalId = "-100";

// set the customForm field on the Customer record to reference the
// form
customer.customForm = customFormRef;

If a form is NOT specified, then the default preferred form for that record is used. If a custom form is
saved with a record in the UI, that form is not used in the SOAP web services request unless it is also the
default form for that record type or is explicitly set as the form in the SOAP web services request.

Sublists in SOAP Web Services

Most records in NetSuite include sublists, which generally contain a list of references to other records.
This figure shows the Expenses sublist on the check record. Notice that this sublist includes two lines in
the Expenses sublist.

Note: For general information on sublists in NetSuite, see the help topic What is a Sublist?

In SOAP web services, sublists are represented in a record's schema definition by its List elements.

Note: Except for the customFieldList element (which contains a record's custom fields), all other
elements that end in List represent a record's sublists.

The following is a portion of the schema for the Check record. (Although the XSD shown below describes
the Items sublist on the check record, the same pattern for defining sublists applies to all sublists on all
records.)

Notice that the sublists on the check record include the Expenses sublist, the Items sublist, and the
Landed Costs sublist (not shown in the figure above because the Landed Costs feature is not enabled in
the account).

```xml
<complexType name="Check">
  <complexContent>
    <extension base="platformCore:Record">
      <sequence>
        <element name="createdDate" type="xsd:dateTime" minOccurs="0"/>
        <element name="address" type="xsd:string" minOccurs="0"/>
        <element name="subsidiary" type="platformCore:RecordRef" minOccurs="0"/>
```
Sublists in SOAP Web Services

The itemList element (listed above) is of type CheckItemList, which is defined in the CheckItem complex type (defined in transactions.bank.xsd). The CheckItem type (shown below) includes all the properties that can be set for the Item sublist on the check record. Also notice the replaceAll attribute in CheckItemList. You will set the value of replaceAll to true or false depending on whether you are working with keyed sublists or non-keyed sublists. See Updating Sublists in SOAP Web Services for details.

```
<complexType name="CheckItem">
    <sequence>
        <element name="item" type="platformCore:RecordRef" minOccurs="0"/>
        <element name="vendorName" type="xsd:string" minOccurs="0"/>
    ....
        <element name="line" type="xsd:long" minOccurs="0"/>
        <element name="location" type="platformCore:RecordRef" minOccurs="0"/>
        <element name="isBillable" type="xsd:boolean" minOccurs="0"/>
        <element name="customFieldList" type="platformCore:CustomFieldList" minOccurs="0"/>
    </sequence>
</complexType>
<complexType name="CheckItemList">
    <sequence>
        <element name="item" type="tranBank:CheckItem" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="replaceAll" type="xsd:boolean" default="true"/>
</complexType>
```

Updating Sublists in SOAP Web Services

When working with sublists, the approach for updating the lines in a sublist will vary depending on whether you are working with Keyed Sublists or Non-Keyed Sublists.

**Keyed Sublists**

Keyed sublists have an indexing line element or internal ID that can be used as a key. Keyed sublists allow you to set the replaceAll attribute to FALSE to update only the lines you are submitting in your SOAP request. When working with keyed sublists, the value of the replaceAll attribute has the following effects:

- replaceAll = TRUE: The existing sublist is **replaced** with the sublist submitted in the SOAP web services request. The newly submitted sublist should include all values.
  - Lines that do not match the newly submitted lines are removed.
  - Currently existing lines that match lines in the new sublist submission are updated.
  - Newly submitted lines with no matches are added.
  - The default value for the replaceAll attribute is TRUE.
replaceAll = FALSE: Lines in the existing sublist are **selectively updated** with lines in the sublist submitted in the SOAP web services request. The newly submitted sublist need only include values to be added and updated.
- Lines that do not match the newly submitted lines are preserved.
- Currently existing lines that match lines in the new sublist submission are updated.
- Newly submitted lines with no matches are added.

The following table lists sublists that are categorized as keyed sublists in SOAP web services. Sublists that do not appear in this table are considered **Non-Keyed Sublists**. The sublist key listed for each keyed sublist below is used to link the sublist data with the appropriate record.

**Important:** PricingMatrixList and demandPlanDetailList are matrix sublists. Updates to matrix sublists may require additional handling related to the replaceAll attribute. See Matrix Sublists and replaceAll.

<table>
<thead>
<tr>
<th>Sublist</th>
<th>Sublist Key</th>
<th>Appears on Record(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>accruedTimeList</td>
<td>payrollitem</td>
<td>Employee</td>
</tr>
<tr>
<td>addressbookList</td>
<td>internalId</td>
<td>Contact, Customer, Employee, Lead, Partner, Project, Prospect, Vendor</td>
</tr>
<tr>
<td>applyList</td>
<td>doc</td>
<td>Customer Payment, Credit Memo, Customer Refund, Customer Deposit, Deposit Application, Vendor Credit, Vendor Payment</td>
</tr>
<tr>
<td>assigneeList</td>
<td>resource</td>
<td>Project Task</td>
</tr>
<tr>
<td>attendeeList</td>
<td>attendee</td>
<td>Calendar Event</td>
</tr>
<tr>
<td>binNumberList</td>
<td>binNumber</td>
<td>Assembly Item (BOM Item), Inventory Item, Lot Numbered Assembly Item, Lot Numbered Inventory Item, Serialized Assembly Item, Serialized Inventory Item</td>
</tr>
<tr>
<td>campaignDirectMailList</td>
<td>internalId</td>
<td>Campaign</td>
</tr>
<tr>
<td>campaignEmailList</td>
<td>internalId</td>
<td>Campaign</td>
</tr>
<tr>
<td>campaignEventList</td>
<td>internalId</td>
<td>Campaign</td>
</tr>
<tr>
<td>companyContributionList</td>
<td>payrollitem</td>
<td>Employee</td>
</tr>
<tr>
<td>componentList</td>
<td>item</td>
<td>Assembly Unbuild, Assembly Build</td>
</tr>
<tr>
<td>contactList</td>
<td>company</td>
<td>Customer, Partner, Vendor</td>
</tr>
<tr>
<td>creditList</td>
<td>doc</td>
<td>Customer Payment, Vendor Payment</td>
</tr>
<tr>
<td>creditCardsList</td>
<td>internalId</td>
<td>Customer, Job</td>
</tr>
<tr>
<td>currencyList</td>
<td>currency</td>
<td>Customer (when Multi-Currency Customer enabled)</td>
</tr>
<tr>
<td>deductionList</td>
<td>payrollitem</td>
<td>Employee</td>
</tr>
<tr>
<td>depositList</td>
<td>doc</td>
<td>Customer Payment, Customer Refund</td>
</tr>
</tbody>
</table>
### Sublists in SOAP Web Services

<table>
<thead>
<tr>
<th>Sublist</th>
<th>Sublist Key</th>
<th>Appears on Record(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>deptAccessList</td>
<td>dept</td>
<td>Custom Entity Field, CRM Custom Field, Other Custom Field, Item Custom Field, Transaction Body Custom Field, Transaction Column Custom Field, Item Option Custom Field, Custom Record Custom Field, Item Number Custom Field</td>
</tr>
<tr>
<td>dimensionList</td>
<td>sequence</td>
<td>Fair Value Price</td>
</tr>
<tr>
<td>directDepositList</td>
<td>internalid</td>
<td>Employee</td>
</tr>
<tr>
<td>earningList</td>
<td>payrollitem</td>
<td>Employee</td>
</tr>
<tr>
<td>expCostList</td>
<td>doc</td>
<td>Invoice, Cash Sale</td>
</tr>
<tr>
<td>expenseList</td>
<td>orderLine or line</td>
<td>Check, Expense Report, Item Receipt, Purchase Order, Purchase Requisition, Vendor Bill, Vendor Credit, Vendor Return Authorization</td>
</tr>
<tr>
<td>itemList</td>
<td>orderLine or line</td>
<td>Bin Putaway Worksheet, Bin Transfer, Cash Sale, Check, Estimate, Inventory Transfer, Invoice, Item Fulfillment, Item Receipt, Credit Memo, Return Authorization, Cash Refund, Opportunity, Purchase Order, Purchase Requisition, Sales Order, Vendor Bill, Vendor Credit, Vendor Return Authorization, Work Order</td>
</tr>
</tbody>
</table>

**Notes:**

- On Item sublists, the value **End of Group** is not returned in SOAP web services. This behavior mimics the behavior of the UI.
- For cash sale records, the orderLine field establishes the relationship with an existing sales order, if any. You must set a value for orderLine to populate the createdFrom field.
- To create a credit memo with the createdFrom field populated, you must set a value for orderLine. In subsequent update requests, you must use “line” for line updates.
- For item receipt records, orderLine is used for transforms, because it implies a link between the previous transaction and the current one. For example, to add an item receipt from a purchase order, the purchase order lines would be "orderLines", because the receipt has not been saved. After the item receipt is saved, lines should be accessed via the "line" field.
- For opportunity records, line is used.
<table>
<thead>
<tr>
<th>Sublist</th>
<th>Sublist Key</th>
<th>Appears on Record(s)</th>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>milestoneList</td>
<td>milestoneId</td>
<td>Billing Schedule</td>
<td>This sublist is available if the Merchandise Hierarchy feature is enabled at Setup &gt; Company &gt; Setup Tasks &gt; Enable Features, on the Items &amp; Inventory subtab.</td>
</tr>
<tr>
<td>nexusesTaxList</td>
<td>nexus</td>
<td>Tax Type</td>
<td>This sublist is available only when scheduleType is set to Fixed Bid Milestone.</td>
</tr>
<tr>
<td>partnersList</td>
<td>partner</td>
<td>Estimate, Cash Sale, Invoice, Sales Order, Opportunity, Credit Memo, Return Authorization, Cash Refund, Customer</td>
<td></td>
</tr>
<tr>
<td>partnersList</td>
<td>partner</td>
<td>Promotion Code</td>
<td></td>
</tr>
<tr>
<td>paymentList</td>
<td></td>
<td>Deposit</td>
<td></td>
</tr>
<tr>
<td>periodDemandPlanList</td>
<td>startDate</td>
<td>Item Demand Plan</td>
<td></td>
</tr>
<tr>
<td>predecessorList</td>
<td>task</td>
<td>Project Task</td>
<td></td>
</tr>
<tr>
<td>pricingMatrixList</td>
<td></td>
<td>Item</td>
<td></td>
</tr>
<tr>
<td>ratesList</td>
<td>subsidiary</td>
<td>Expense Category</td>
<td>Sublist available only to OneWorld accounts.</td>
</tr>
<tr>
<td>recurrence</td>
<td>recurrenceId</td>
<td>Billing Schedule</td>
<td></td>
</tr>
<tr>
<td>resourceList</td>
<td>resource</td>
<td>Calendar Event</td>
<td></td>
</tr>
<tr>
<td>roleAccessList</td>
<td>role</td>
<td>Entity Custom Field, CRM Custom Field, Other Custom Field, Item Custom Field, Transaction Body Custom Field, Transaction Column Custom Field, Item Option Custom Field, Custom Record Custom Field, Item Number Custom Field</td>
<td></td>
</tr>
<tr>
<td>salesTeam</td>
<td>employee</td>
<td>Customer, Invoice, Cash Sale</td>
<td></td>
</tr>
<tr>
<td>taxRegistrations</td>
<td>id</td>
<td>Customer, Partner, Vendor</td>
<td></td>
</tr>
</tbody>
</table>

Sublist available when the SuiteTax feature is enabled.
### Example

The following sample shows how to add an item to an opportunity record and modify an existing item in that record. To do this, set replaceAll to FALSE, add the new item, and modify the existing item by referencing the desired item in the line property. The following shows that the existing line item is modified to change the quantity.

```
<soapenv:Body>
<platformMgs:update xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/
xmlns:s0="urn:sales_2017_1.transactions.webservices.netsuite.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:platformMsgs="urn:messages_2017_1.platform.webservices.netsuite.com">
<platformMsgs:record xsi:type="s0:Opportunity" internalId="1638">
 <s0:itemList replaceAll="false">
   <s0:item>
     <s0:item internalId="380" type="inventoryItem" />
     <s0:quantity>1.0</s0:quantity>
     <s0:amount>20.0</s0:amount>
   </s0:item>
   <s0:item>
     <s0:line>2</s0:line>
     <s0:quantity>10.0</s0:quantity>
   </s0:item>
 </s0:itemList>
</platformMsgs:record>
</platformMgs:update>
</soapenv:Body>
```

### Matrix Sublists and replaceAll

Matrix sublists respect the replaceAll attribute for updates, as other Keyed Sublists do, but in some cases, updates to matrix sublists require additional handling.

Matrix sublists require additional handling for replaceAll because body-level fields slave values into them. For example, setting the pricing schedule for an item dictates values in the pricing matrix. In the UI, when a pricing schedule is selected, the pricing schedule logic is run first, and values specified in the pricing matrix sublist are taken into account to arrive at what the update should do. However, the UI does not have the notion of replaceAll because the user interacting with the form can delete values they do not want to keep in individual fields.

SOAP web services requests cannot delete individual field values in a sublist. The only way to get rid of old values is to set replaceAll to TRUE and send all values in the update request. Some body fields can override values in a matrix sublist, but a SOAP web services request can then change some or all of these sublist values, depending on the replaceAll setting.

Review the following cases to get an understanding of how the slaving of values from body fields into a matrix sublist interacts with the replaceAll setting on updates:

- No sublist values from body fields and replaceAll=TRUE
The SOAP web services request does not include any body field values that drive values in the sublist, so the update is handled like those in other keyed sublists where replaceAll is set to TRUE.

The existing sublist is replaced with the sublist submitted in the SOAP web services request. The newly submitted sublist should include all values.

- No sublist values from body fields and replaceAll = FALSE
  The SOAP web services request does not include any body field values that drive values in the sublist, so the update is handled like those in other keyed sublists where replaceAll is set to FALSE.
  Lines in the existing sublist are selectively updated with lines in the sublist submitted in the SOAP web services request. The newly submitted sublist need only include values to be added and updated.

- Sublist values from body fields and replaceAll=TRUE
  The SOAP web services request sets body field value(s) that drive values in the sublist, so specialized handling is required.
  First, all sublist values are removed. Next, sublist values driven by body field values are set. Then, the sublist is updated with all lines specified in the newly submitted sublist. The newly submitted sublist lines thus override the values driven by body fields.

**Important:** When sublist values from a SOAP web services request override values driven by body fields, errors may occur. If you do not want these overrides, you must set replaceAll to FALSE. For example, if you are setting a quantity pricing schedule, you should set ReplaceAll to FALSE.

- Sublist values from bodyfields and replaceAll=FALSE
  The SOAP web services request sets body field value(s) that drive values in the sublist, so specialized handling is required.
  First, sublist values driven by body field values are set. Then, the sublist is updated with any updated or added lines from the newly submitted sublist.

**Note:** By default, replaceAll is set to TRUE for matrix sublists, like other sublists.

**Currently Supported Matrix Sublists**

The pricingMatrixList on Items and periodDemandPlanList on Item Demand Plans are currently supported matrix sublists. For details about these sublists, see the help topics Pricing Matrix List and Item Demand Plan.

**Important:** In endpoints prior to 2012.1, updates to pricingMatrixList ignore the replaceAll attribute. For the 2012.1 and later endpoints, replaceAll is respected for this sublist.

**Non-Keyed Sublists**

Non-keyed sublists have no indexing line element or internal ID that can be used as a key. Each line in a non-keyed sublist is recorded in a flat list.

In the UI, non-keyed sublists and keyed sublists may look similar; however, technically, they are quite different. Because non-keyed sublists contain no referencing keys (or handles), you cannot update a specific line in a non-keyed sublist. Instead, you must interact with the sublist as a whole. In non-keyed sublists, the replaceAll attribute is ignored and behaves as if it were set to TRUE for all requests. Consequently, an update operation is similar to the add operation with respect to non-keyed sublists.
In the context of SOAP web services, all sublists not listed as Keyed Sublists are considered to be non-keyed sublists. The following bullets outline the behaviors of non-keyed sublists.

- To update a single line in the sublist, retrieve the entire sublist, change a single line as desired, and then resubmit the entire sublist.
- To replace the sublist with a subset of lines, retrieve the entire sublist, and then resubmit the subset of lines. The original sublist is purged and replaced by the new sublist containing the subset of lines.
- To delete the sublist, submit an empty sublist. See Deleting All Lines on a Sublist for more information.

Sublist Line Numbers

The index number for each sublist line is assigned by the system. SOAP web services developers cannot change these values. When lines are added or removed from a sublist (using either the UI or SOAP web services), the system does not re-index the line numbers. Therefore, it is best practice to perform a get operation on the sublist data before trying to update individual lines, otherwise you may be updating the wrong line in the sublist.

For example, if you perform a get operation on an Items sublist on a transaction record, the index values for the first three line items might be:

```xml
<ns3:line>1</ns3:line>
<ns3:line>8</ns3:line>
<ns3:line>9</ns3:line>
```

Deleting All Lines on a Sublist

This sample shows how to delete all addresses from the Address sublist on a customer record (CustomerAddressbookList).

**Java**

```java
Customer update = new Customer();
update.setInternalId(c.getInternalId());
cabl = new CustomerAddressbookList();
cabl.setAddressbook(new CustomerAddressbook[0]);
cabl.setReplaceAll(true);
update.setAddressbookList(cabl);
```

**SOAP**

```xml
<update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <record internalId="724" xsi:type="ns1:Customer"
  xmlns:ns1="urn:relationships_2017_1.lists.webservices.netsuite.com">
    <ns1:addressbookList replaceAll="true" xsi:type="ns1:CustomerAddressbookList"/>
  </record>
</update>
```

The next sample shows how to delete all line items from the Item sublist (OpportunityItemList) on an opportunity record.
Java

```java
Opportunity update = new Opportunity();
update.setInternalId(opp.getInternalId());
update.setItemList(new OpportunityItemList(new OpportunityItem[0], true));
```

SOAP

```xml
<update xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <record internalId="6147" xsi:type="ns1:Opportunity"
           xmlns:ns1="urn:sales_2017_1.transactions.webservices.netsuite.com">
    <ns1:itemList replaceAll="true" xsi:type="ns1:OpportunityItemList"/>
  </record>
</update>
```

Searching a Sublist

Keyed sublists have a **line** field (typically on transaction sublists). Individual lines in the list can be searched for by referencing the line value.

Also note that when searching records, by default sublists are NOT returned in the result set, which increases the search response time. However, if you want to return all sublist data in your search, set the `SearchPreferences.bodyFieldsOnly` preferences to FALSE in your search request.
SOAP Web Services Processing

This section describes synchronous and asynchronous processing. Also provided are steps for monitoring your SOAP web services requests. Refer to the following topics:

- Synchronous Versus Asynchronous Request Processing
- Asynchronous Request Processing
- Synchronous Request Processing

Synchronous Versus Asynchronous Request Processing

SOAP web services requests can be processed synchronously or asynchronously.

With synchronous requests, your client application sends a request to NetSuite, and the client waits until the request is processed and a response is returned. That is, the application does not proceed with other work until receiving the response.

With asynchronous requests, your client application sends a request to SOAP web services, where it is placed in a processing queue and handled asynchronously with other requests. Your client application does not wait for a response but goes on to other work. After a job is submitted, a job Id is returned in the SOAP web services response. Your client application can then check on the status and result of the request by referencing the job Id.

Asynchronous processing may be advantageous in the following situations:

- If you expect your connection to NetSuite to be slow or unstable.
- If your job is large, and its processing can be postponed until off-peak hours.

Be aware that asynchronous responses may not be returned immediately. Before committing to using asynchronous operations, you should consider this factor and decide whether it fits in with your business logic.

Asynchronous Request Processing

This section describes asynchronous request processing in more detail.

Available Operations

Asynchronous equivalents are available for several synchronous operations, as shown in the table below.

<table>
<thead>
<tr>
<th>Asynchronous Operation</th>
<th>Synchronous Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>asyncAddList</td>
<td>addList</td>
</tr>
<tr>
<td>asyncUpdateList</td>
<td>updateList</td>
</tr>
</tbody>
</table>
### Asynchronous Operation vs. Synchronous Equivalent

<table>
<thead>
<tr>
<th>Asynchronous Operation</th>
<th>Synchronous Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>asyncUpsertList</td>
<td>upsertList</td>
</tr>
<tr>
<td>asyncDeleteList</td>
<td>deleteList</td>
</tr>
<tr>
<td>asyncGetList</td>
<td>getList</td>
</tr>
<tr>
<td>asyncSearch</td>
<td>search</td>
</tr>
<tr>
<td>asyncInitializeList</td>
<td>initializeList</td>
</tr>
</tbody>
</table>

### Understanding Asynchronous Requests and Responses

When submitting an asynchronous request, the request is similar to the equivalent synchronous call. The following code illustrates an asynchronous request for a list of customer records.

```xml
<soap:Body>
<platformMsgs:asyncGetList>
<platformMsgs:baseRef internalId="87" type="customer" xsi:type="platformCore:RecordRef"></platformMsgs:baseRef>
<platformMsgs:baseRef internalId="176" type="customer" xsi:type="platformCore:RecordRef"></platformMsgs:baseRef>
</platformMsgs:asyncGetList>
</soap:Body>
```

The response, however differs in that only information about the request is returned. This information includes a unique identifier for the request, which is enclosed in the `jobId` tags.

```xml
<soapenv:Body>
<asyncGetListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<asyncStatusResult xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
<jobId>ASYNCWEBSERVICES_563214_0531201943428686160042948_4bee0685</jobId>
<status>pending</status>
<percentCompleted>0.0</percentCompleted>
<estRemainingDuration>0.0</estRemainingDuration>
</asyncStatusResult>
</asyncGetListResponse>
</soapenv:Body>
```

Later, you can use the `checkAsyncStatus` or `getAsyncResult` operations to track the asynchronous request. To identify the specific request about which you need data, refer to it by its `jobId`.

### Usage Notes

Be aware of the following characteristics of asynchronous processing:

- All available jobs are processed contiguously. That is, processing priority is based on when the job is submitted, and jobs cannot pass one another in the queue.
- The system does not enforce waiting periods for jobs. That is, as soon as a job is completed, the next job from the queue is immediately selected and the processing starts.
- Asynchronous request `jobId`s are valid for 30 days. After 30 days, you can no longer look up the request.
Asynchronous Request Processing

Checking the Status of an Asynchronous Job

To check the status of an asynchronous SOAP web services job, you use the checkAsyncStatus operation. When you use this operation, you reference a job Id. In response, the system returns the job's status, among other details.

Specific status values that can be returned include the following:

- failed
- finishedWithErrors — indicates that at least one record was processed, but at least one failed.
- pending
- processing
- finished

If the status is failed, finishedWithErrors, or finished, you can in most cases use the getAsyncResult operation to obtain more details. These details may include the record results, or error and fault messages. The getAsyncResult operation is described in Checking for Detailed Results of an Asynchronous Job.

checkAsyncStatus also returns details about the percentage of the job that is complete, and the estimated time remaining. The estimated remaining time is measured in seconds.

checkAsyncStatus Request and Response

The following is a typical checkAsyncStatus request.

```xml
<soap:Body>
    <platformMsgs:checkAsyncStatus>
       <platformMsgs:jobId>ASYNCWEBSERVICES_563214_053120061943428686160042948_4bee0685</platformMsgs:jobId>
    </platformMsgs:checkAsyncStatus>
</soap:Body>
```

The following is a typical checkAsyncStatus response.

```xml
<soapenv:Body>
    <checkAsyncStatusResponse
        xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
        <asyncStatusResult xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
            <jobId>ASYNCWEBSERVICES_563214_053120061943428686160042948_4bee0685</jobId>
            <status>pending</status>
            <percentCompleted>0.0</percentCompleted>
            <estRemainingDuration>0.0</estRemainingDuration>
        </asyncStatusResult>
    </checkAsyncStatusResponse>
</soapenv:Body>
```

checkAsyncStatus Faults

The checkAsyncStatus operation can throw any of the following faults:

- InvalidSessionFault
Asynchronous Request Processing

- AsyncFault
- UnexpectedErrorFault

For more information on faults, see SOAP Fault Status Codes.

Checking for Detailed Results of an Asynchronous Job

To retrieve details about specific records submitted as part of an asynchronous request, you use the getAsyncResult operation.

**getAsyncResult Time Limits**

Be aware of the following limits:

- You can use the getAsyncResult operation for a particular job ID up to 20 times within a 30-day time period. If this limit is exceeded, the system returns an INVALID_JOBID user error.
- Async job IDs are purged from the system every 30 days. If you attempt to download the results 31 days after your initial request, the system returns an INVALID_JOBID user error.

**getAsyncResult Example**

You can use code like the following to retrieve the getAsyncResult response. This example shows how to look for records that failed with a status detail code of "unexpected error" and then take some action with those records.

```java
AsyncAddListResult result = (AsyncAddListResult)port.getAsyncResult("job ID", 1);
WriteResponseList responses = result.getWriteResponseList();
for (WriteResponse response : responses.getWriteResponse())
{
    if (!response.getStatus().isIsSuccess())
    {
        if (response.getStatus().getStatusDetail()[0].getCode() == StatusDetailCodeType.UNEXPECTED_ERROR)
            // add for resubmit
    }
}
```

**Note:** To use getAsyncResult with asyncInitializeList, you must be using the 2013.2 endpoint or later. You can monitor asyncInitializeList requests using the UI, as described in Monitoring Asynchronous Jobs from the UI.

**getAsyncResult Faults**

This operation can throw one of the following faults.

- InvalidSessionFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- AsyncFault
Asynchronous Request Processing

- UnexpectedErrorFault

For more information on faults, see SOAP Fault Status Codes.

**Note:** You can see some details about asynchronous job results by navigating to Setup > Integration > SOAP Web Services Process Status and viewing the Response.xml file for the job. For details, see Monitoring Asynchronous Jobs from the UI.

Monitoring Asynchronous Jobs from the UI

SOAP web services jobs submitted asynchronously can be monitored at Setup > Integration > SOAP Web Services Process Status. To access this log, either the full Web Services permission or the View Web Services Logs permission is needed. Both permissions are available on the role record's Permissions subtab under Setup.

On this page, you can view the following information about each job:

- **Date** — the date the job was created
- **Sequence** — the order of the job in relation to other asynchronous jobs in process (jobs are processed on a first in, first out basis)
- **Job Number** — order number of the job in the work queue
- **Job Id** — the Job Id, which is also returned in the asynch SOAP response, and can be used to programmatically retrieve job status information
- **Status** — the status of the job: failed, finishedWithErrors, pending, processing, or finished (for more on status, see Checking the Status of an Asynchronous Job)
- **Percent Complete** — the percentage of job processing that has been completed
- **Est. Remaining Time** — the estimated amount of time remaining before the job is completed
- **Request** — the SOAP request associated with the job (see additional details below)
- **Response** — the SOAP response associated with the job (see additional details below)
- **Cancel** — you can check the box in this column to cancel a job that has not yet successfully completed

Limits on Requests and Responses

Note the following limitations:

- A response for a job is available only after the job has successfully completed.
- In production environments, the data is accessible for 21 days. In sandbox environments, the data is accessible for seven days.
- Lengthy requests and responses are not saved by the system and therefore are not available for your review. The system imposes the following limits:
  - The system does not save requests that are larger than 100MB.
  - The system does not save responses that are large than 1MB.

Refreshing the Job Status Page

The Job Status page does not automatically refresh. To get the current status of a job, click the Refresh button at the top of the page.
Synchronous Request Processing

In addition to using the responses NetSuite sends for each request, you can monitor synchronous SOAP web services processing in the NetSuite user interface. You have the following options:

- Referring to the execution log of the appropriate integration record. For details on working with integration records, see Integration Management.
- Using the SOAP Web Services Usage Log
- Creating Integration Reports

See also Handling of Lengthy Requests, which describes limitations that exist for sending responses.

For information on monitoring asynchronous requests, see Monitoring Asynchronous Jobs from the UI.

Using the SOAP Web Services Usage Log

Use the SOAP Web Services Usage Log to monitor synchronous requests. This page can be accessed at Setup > Integration > SOAP Web Services Usage Log.

Only administrators can access the SOAP Web Services Usage Log. Assigning the Web Services permission to a role does not provide access to this log. Access is provided to administrators only because access to this log could allow users to see data they do not have permission to view.

The SOAP Web Services Usage Log allows users to filter their SOAP web services requests by specified date and time periods, record type, and action (operation). The SOAP requests and responses for each job are accessible in the Request and Response columns. In production environments, requests and responses are accessible for 21 days.

**Note:** In case of heavy SOAP web services activity, the logs might be purged earlier than 21 days.

In sandbox environments, requests and responses are accessible for seven days. In sandbox environments, you can view request and response information in log and search results after seven days. However, links to the original SOAP request and responses are not available.

Results can be filtered down to the minute. The results returned provide second-level granularity.

Note that lengthy requests and responses are not saved by the system and therefore are not available for your review. The system imposes the following limits:

- The system does not save requests that are larger than 100 MB.
- The system does not save responses that are larger than 10 MB.

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

- The request job ID.
- The date and time that the request was made.
- The duration of the request.
- Information about the requests that were rejected due to a violation of the concurrency governance.
- The integration record used for the request.
- The type of action used in the request.
- The record type that was the subject of the request.
Synchronous Request Processing

- The email address of the user who sent the request.
- The status of the request.
- The number of records included in the request.
- The number of request records that have finished.
- The number of request records that failed.
- The number of records returned in the request’s query.
- Links to the SOAP request and response.

**Note:** The SOAP Web Services Usage Log does not show asynchronous requests. For details, see Monitoring Asynchronous Jobs from the UI.

Creating Integration Reports

You can monitor SOAP web services processing by creating the following types of integration reports:

- Integration and Automation Usage Summary by Job
- Integration and Automation Usage Summary by Record Type

**Important:** The Integration permission is required to view these reports. To enable this permission, administrators must go to Setup > Users/Roles > Manage Roles. Click Customize next to the appropriate role. On the Permissions tab, click Reports. In the Permissions dropdown list, select Integration and click Add.

**Note:** You might receive a timeout error when creating integration reports for accounts with large volumes of SOAP web services traffic. Also, although integration reports do offer some filtering capabilities, users must first take the added step of customizing the report. Be aware that you can apply filtering in the SOAP Web Services Usage Log without doing customization.

Both of these reports provide details on type of operations used (for example, add, addList, delete), who performed an operation, and the time an operation was performed.

Web Services Integration reports are also useful for administrators who need to diagnose and troubleshoot issues. Note that each report can be customized to display only the desired information. To customize a report, click Customize next to the desired report and then modify the report as desired. For details about how to customize a report, see the help topic Report Customization.

If an integration report is timing out, it is recommended that you customize the report by adding filters so the report returns fewer results. See Filtering Integration Report Data.

**Note:** These reports do not support reporting by period even when the Report by Period preference is set to All Reports. The Report by Period preference can be configured at Home > Set Preferences, the Analytics subtab.

Integration and Automation Usage Summary by Job

This report can be used for performance statistics. It shows the duration of each SOAP web services processing job, the number of records modified in each job, the number of successful versus failed record
modifications for each job, and the number of records queried for each job. You can also drill down to job details by clicking the job number.

Any related SOAP files are stored in View links in the reports Request Doc and Response Doc columns. Be aware that NetSuite purges SOAP files every 30 days. In sandbox environments, these files are accessible for seven days.

**Note:** In sandbox environments, you can view request and response information in log and search results after 7 days. However, links to the original SOAP request and responses are not available.

You can view this report at Reports > Integration > Integration and Automation Usage Summary by Job.

**Integration and Automation Usage Summary by Record Type**

This report lists the record types modified by SOAP web services processing, and the number of records added, updated, deleted, and queried for each record type.

You can view this report at Reports > Integration > Integration and Automation Usage Summary by Record Type.

**Filtering Integration Report Data**

You can customize integration reports to run faster, by adding filters that limit the returned results. For example, you could limit a report to return only jobs with start dates that fall into a specified date range.

**To filter an integration report by start date:**

1. Go to Reports > Integration > Integration and Automation Usage Summary by Job > Customize or Reports > Integration > Integration and Automation Usage Summary by Record Type > Customize.
2. Click Filters.
3. Under Search Fields box, enter Start Date and click the Search button.
   The Actual Job Start Date and Start Date fields are listed.
4. Click Start Date.
   This field is added to the Choose Filters pane.
5. In the Choose Filters pane:
   1. Set Filter to between.
   2. Set Date Range to custom.
   3. In the From and To fields, enter dates in (mm/dd/yyyy) format.
   4. Click Done.
6. Click Save.

**Handling of Lengthy Requests**

Throughout NetSuite, the maximum duration for a single HTTP connection is 15 minutes. However, in some cases a SOAP web services request may require more than 15 minutes to process. Because the
system cannot send a response after the connection has timed out, the system handles lengthy requests differently from those that complete more quickly. The exact handling varies depending on the operation. For details, see the following:

- Single-Record Requests
- List and Search Requests

### Single-Record Requests

If a synchronous single-record operation does not complete within 15 minutes, the operation is allowed to finish. In these cases, no response is sent, but the system updates the SOAP Web Services Usage Log with a partial response. However, note that single-record operations almost always complete before the HTTP connection times out.

A single-record operation is one that creates or affects only one record. For example, add, update, and delete are single-record operations.

### List and Search Requests

In general, a list or search request that requires longer than 15 minutes to process is terminated by the system. The exact behavior varies depending on the endpoint.

### 2014.2 Endpoint

When the 2014.2 or a later endpoint is being used, NetSuite sends a response 10 seconds prior to the connection timeout. The operation is terminated after the record currently being processed has finished. The system uses the messaging described in the following table.

<table>
<thead>
<tr>
<th>Operation Type</th>
<th>Relevant Operations</th>
<th>Messaging</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search</td>
<td>search, searchMore, searchMoreWithId, searchNext</td>
<td>The response indicates that the operation did not finish because it timed out.</td>
</tr>
<tr>
<td>List</td>
<td>addList, deleteList, getList, initializeList (for details, see initialize / initializeList), updateList, upsertList</td>
<td>The response indicates that the operation did not finish because it timed out. For records that were successfully processed, the response indicates success for that record, along with other standard details, such as the record's internal ID. For other records, the status indicates one of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ The record's status is unknown - This status is used if the processing of the record was still in progress when the response was being created.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ The record was not processed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Using this information and the original SOAP request, you can determine which data to resend.</td>
</tr>
</tbody>
</table>

The following screenshot shows a response for an addList operation that was terminated.
2014.1 Endpoint and Earlier

If you are using 2014.1 or an earlier endpoint, a lengthy operation is stopped after 15 minutes. The system does not send a response, but it updates the SOAP Web Services Usage Log with a partial response, which shows only the records that were processed. For help with the SOAP Web Services Usage Log, see Using the SOAP Web Services Usage Log.
SOAP Web Services Security

NetSuite leverages the latest industry security standards to ensure high levels of security around your business data. All SOAP web services requests are controlled by the same security measures used in the NetSuite UI. This includes:

- Authentication for SOAP Web Services
- Authorization for SOAP Web Services
- Session Management for SOAP Web Services
- Encryption for SOAP Web Services

Additional security-related topics include:

- Custom Field Security - for information on security applied at the field level and its affect on SOAP web services
- PCI Compliance Password Requirements - for information on password requirements associated with the Payment Card Industry (PCI) Data Security Standard

Authentication for SOAP Web Services

Authentication is the process of determining the identity of requesters by verifying the credentials they present. Most SOAP web services operations require authentication.

The available methods of authentication include the following:

- Token-Based Authentication
- User Credentials
- Outbound Single Sign-on (SuiteSignOn)
- Inbound Single SignOn (Inbound SSO)

**Important:** Use only one authentication method in one SOAP message. Mixing different authentication types in a single SOAP message returns a SOAP fault.

Token-Based Authentication

With this approach, the request must include a consumer key, a token, and other data. Authentication details must be sent by using request-level credentials, not the login operation. For details on structuring a SOAP request this way, see Request-Level Credentials.

You generate a consumer key and secret when you create an integration record and configure the record to permit token-based authentication (by checking the Token-based Authentication box). For details, see Integration Management.

**Important:** Token-based authentication is supported for use only with the 2015.2 endpoint and later.

User Credentials

With this approach, the request must include the credentials associated with a single user: an email address, password, role, and NetSuite account ID. To find your account ID, go to the SOAP Web Services
Preferences page at Setup > Integration > Integration Management > SOAP Web Services Preferences. For a list of NetSuite role internal IDs, see Internal IDs Associated with Roles.

When you authenticate by using user credentials, there are two approaches to how the SOAP request may be structured: you can use the login operation or you can use request-level credentials. For details, see Structuring SOAP Requests to Include Authentication Details.

For requests that use the 2015.2 WSDL or later, application ID is also mandatory, although technically application ID is not considered a part of the user authentication process. An application ID is used to link a request to an integration record that exists in your NetSuite account. To use any integration record for requests that authenticate through user credentials, the record must have the User Credentials box checked. For details, see Integration Management.

**Outbound Single Sign-on (SuiteSignOn)**

This approach allows an external application to perform callbacks to NetSuite after the SuiteSignOn handshake is complete and there is a live SuiteSignOn token. See the SuiteSignOn Sequence Diagram and Connection Details for details.

For more information about SuiteSignOn, see the help topic Outbound Single Sign-on (SuiteSignOn).

**Inbound Single SignOn (Inbound SSO)**

**Important:** As of 2018.2, new Inbound Single Sign-on (SSO) solutions are no longer enabled for use with SOAP web services for new customers or new partners. Current partners may continue to sell their existing Inbound SSO solutions to new customers. However, it is strongly recommended that you consider other options, and use token-based authentication (TBA) if possible.

Inbound SSO is used to provide authentication to NetSuite on behalf of an individual who has already authenticated to an external application.

With inbound SSO, a mapping is created between a user’s NetSuite account and the user’s external account. The mapping can be created by using the mapSso operation, or it can be created manually in the UI. After the mapping is created, the user can be automatically logged in to NetSuite by using the ssoLogin operation, which is described in ssoLogin. The ssoLogin operation requires values representing the partner, the user’s ID in the partner system, and other details.

Note that the ssoLogin operation functions in much the same way as the login operation. For more details, see Login and ssoLogin Operations.

For mapSso and ssoLogin requests that use the 2015.2 WSDL or later, application ID is mandatory, although technically application ID is not considered a part of the authentication process. An application ID is used to link a request to an integration record that exists in your NetSuite account. To use an integration record for requests that authenticate through inbound SSO, the record must have the User Credentials box checked. For details, see Integration Management.

**Structuring SOAP Requests to Include Authentication Details**

If your chosen authentication method is user credentials, the SOAP web services platform supports two approaches for structuring your SOAP requests. These approaches are described in the following table.
If your authentication method is **inbound SSO**, you must use the first style of authentication listed below. For **token-based authentication**, you must use request-level credentials.

<table>
<thead>
<tr>
<th>Name</th>
<th>Valid with</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login/ssoLogin operation</td>
<td>Authentication through user credentials</td>
<td>This approach initiates a session that is governed by timeout and user limits. For details, see Login and ssoLogin Operations.</td>
</tr>
<tr>
<td></td>
<td>Inbound SSO</td>
<td></td>
</tr>
<tr>
<td>Request-level credentials</td>
<td>Authentication through user credentials</td>
<td>You can send credentials in the SOAP header of every request. This approach initiates a stateless form of SOAP web services communication in which the login operation is not invoked and certain timeouts do not apply. For details, see Request-Level Credentials.</td>
</tr>
<tr>
<td></td>
<td>Token-based authentication</td>
<td>For information about using the TokenPassport complex type with token-based authentication, see TokenPassport Complex Type and Updating an Integration to Send Token-Based Authentication Details.</td>
</tr>
</tbody>
</table>

**Note:** If you are using outbound SSO, the information in this topic does not apply. For details on outbound SSO, see the help topic Outbound Single Sign-on (SuiteSignOn).

### Login and ssoLogin Operations

When you use the login operation, typically you would create a login class that includes a Passport object. Then you would call the login operation, passing the passport object as the sole argument. This is how the process works for user credentials.

With inbound SSO, the process is the same: you would create an ssoPassport object. Then you call the ssoLogin operation, passing the ssoPassport object as the sole argument.

The following C# snippet shows how to create a login class for a request that uses user credentials.

```csharp
private void login()
{
    _service.CookieContainer = new CookieContainer();

    Passport passport = new Passport();
    RecordRef role = new RecordRef();

    passport.email = decrypt(myEncryptedEmail);
    passport.password = decrypt(myEncryptedPassword);
    passport.role = new RecordRef { internalId = "3" };  
    passport.account = "1234567";

    SessionResponse resp = _service.login(passport);
}
```
After the requester has been successfully authenticated, a new session is created for that user. When using the login or ssoLogin operation to authenticate to NetSuite, user credentials are stored in HTTP headers and a JSESSIONID is assigned to every session.

If you are using inbound SSO, note also that the mapSSO operation has similar behavior. That is, although the primary purpose of the mapSSO operation is to create a mapping, it also logs the user in for a session. As part of this behavior, it returns a JSESSIONID.

If you are using application IDs, see Using Application ID with the Login and ssoLogin Operations.

**Important:** For session information to be successfully conveyed through a SOAP request, you must enable support for multiple cookie management in your application. For example, when using C#, you would include the following line:

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

**Request-Level Credentials**

Rather than authenticating to NetSuite by invoking the login operation, users have the option of sending their credentials in the SOAP header of each request. Sending credentials with each request eliminates the need for session management and separate logins. This approach may be beneficial to developers using PHP or other scripting languages that do not have built-in mechanisms for session management, manipulating HTTP headers, or tracking session IDs. Best practices to avoid concurrency errors include the use of request-level credentials, along with retries upon concurrency errors.

**Note:** You can use request-level credentials only when authenticating through user credentials or token-based authentication.

Note the following about using request-level credentials:

- To avoid errors, do not accept cookies when you use request-level credentials.
  - To prevent the acceptance of cookies in Java: use setMaintainSession(false).
  - To prevent the acceptance of cookies in C#, do not create a CookieContainer.
- Users must submit a role with every request.
- As with any request, only one SOAP web services request may be active at one time. Any attempt to send a second request results in a fault.

One way to use request-level credentials is to complete the following steps:

1. Assume that you have a class member called _service, which is an instance of the NetSuite service. For example:

   ```csharp
   namespace NetSuite
   {
   
   class mainIntegration
   {
   
   public NetSuiteService _service;
   ...
   ```
2. Create a passport method. The method should create a passport object that sets the user credential fields appropriately. It should also link this object to the _service object. For example:

```csharp
private void setPassport()
{
    Passport passport = new Passport
    {
        email = decrypt(myEncryptedEmail);
        password = decrypt(myEncryptedEmail);
        account = "1234567",
        role = new RecordRef { internalId = "3" } 
    };
    Console.WriteLine("Login info...");
    Console.WriteLine(" Email : {0}, passport.email);
    Console.WriteLine(" Role Internal Id : {0}, passport.role.internalId);
    Console.WriteLine(" Account : {0}\n", passport.account);
    _service.passport = passport;
}
```

The previous example code lists the role internal ID of the user and the account ID. To find your account ID, go to the SOAP Web Services Preferences page at Setup > Integration > SOAP Web Services Preferences. For a list of NetSuite role internal IDs, see Internal IDs Associated with Roles.

3. Use your passport method to modify the SOAP header that will be generated.

```csharp
public SuiteTalkCourse()
{
    ServicePointManager.ServerCertificateValidationCallback += delegate { return true; };
    _service = new DataCenterAwareNetSuiteService();
    setPreferences();
    setApplicationInfo();
    setPassport();
}
```

4. The next time the application sends a request, the passport details are included in the header.

```xml
<soap:Header>
    <applicationInfo xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
        <applicationId>************************************</applicationId>
    </applicationInfo>
    <passport xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
        <email xmlns="urn:core_2017_1.platform.webservices.netsuite.com">my@email.com</email>
        <password xmlns="urn:core_2017_1.platform.webservices.netsuite.com">**********</password>
        <account xmlns="urn:core_2017_1.platform.webservices.netsuite.com">1234567</account>
        <role internalId="3" xmlns="urn:core_2017_1.platform.webservices.netsuite.com">1234567</role>
    </passport>
    <preferences xmlns="urn:messages_2017_1.platform.webservices.netsuite.com"/>
</soap:Header>
```
Authentication for SOAP Web Services

Note: If some classes from the NetSuite WSDL are not generated, set the all parameter in your axis-wsdl2java ant task to true. For more information, see SOAP Web Services FAQ.

Important: If you use request-level credentials in a search request, you must use the searchMoreWithId operation to paginate through search results. See searchMoreWithId for more information.

Using the Login Audit Trail

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

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

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

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

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

The following messages indicate some of the most common issues:

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

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

For more information about error messages displayed in the Login Audit Trail, see the help topic TBA-Related Error Messages in the Login Audit Trail.

Authorization for SOAP Web Services

Authorization is the process of ensuring that the requester has the appropriate entitlement to perform the requested operation. When users request to be authenticated, they also provide their NetSuite role. For every SOAP web services request, the system uses the role definition to ensure that the user has the required permission for the requested operation as well as the requested record type. The role must be provided in the Passport type via the login operation:

Example

```xml
<soapenv:Body>
  <login xmlns="urn:messages_2018_1.platform.webservices.netsuite.com">
```

Authorization for SOAP Web Services

For detailed information on NetSuite roles and permissions and how SOAP web services implement roles and permissions rules, refer to Roles and Permissions in SOAP Web Services.

Session Management for SOAP Web Services

After you have been successfully authenticated using the login operation, a sessionID is created that must be passed to each subsequent request. Additional logins are not required as long as the session is active.

In NetSuite, session management includes:

- Session Timeouts
- Session Limits
- Manually Managing Cookies

⚠️ Important: The following topics pertain to sessions that have been initiated using the login operation. If you authenticate to NetSuite by submitting your credentials in the SOAP header of your request should see Request-Level Credentials.

Session Timeouts

The Open Web Application Security Project (OWASP) provides the following guideline: All sessions should implement an absolute timeout, regardless of session activity. This timeout defines the maximum amount of time a session can be active. The session is closed and invalidated upon the defined absolute period. After the session is invalidated, the user must authenticate (log in) again in the web application and establish a new session. The absolute session timeout limits the amount of time possible for a potential attacker to use a hijacked session to impersonate a user.

⚠️ Important: To enhance the security of your account, and to comply with the OWASP guideline, there is an absolute session timeout implemented for SOAP web services sessions. Currently, the absolute session timeout value is set to 60 minutes, even if the session is active.

If you use sessions with your SOAP web services integrations, you must ensure that your SOAP web services calls are able to handle session timeouts. It is recommended that your integrations should use sessionless protocols based on request-level credentials, such as user credentials or token-based authentication (TBA). Using TBA is recommended. See Authentication for SOAP Web Services for more information.

SOAP web services sessions automatically time out after 20 minutes of inactivity, requiring a login to resume activity. If the server resubmits the cookie during the first 20 minutes, the inactivity timer is reset.
If a SOAP web services operation takes more than 15 minutes to complete, it automatically times out. In this case, consider using asynchronous request processing. For more information, see Synchronous Versus Asynchronous Request Processing.

**Note:** For security reasons, you should always send a logout operation as soon as your business operations are complete. Send a logout operation before the absolute session timeout or the inactivity session timeout.

If you explicitly log out of a session, and then attempt to use the same session, a SESSION_TIMED_OUT error message is returned. Your code should be prepared to handle session timeouts by retrying if an InvalidSessionFault (SESSION_TIMED_OUT) error is received.

**Important:** Do not make any integration decisions based on the value of the absolute timeout value or the idle session timeout values, as these values can be changed at any time without notice.

Session Limits

Session limits involve both user limits and account limits.

**User Limits**

A specific login (user name, account and password) that is used for a UI session can be used to create one or more sessions through SOAP web services by using the login operation. More SOAP web services sessions can occur concurrently, but the number of operations processed concurrently is limited. The UI session does not affect the SOAP web services sessions and operations.

Without a SuiteCloud Plus license, it is recommended for one user to only create one SOAP web services session using the login operation, and send requests using this session in one thread sequentially.

If the NetSuite server receives two login or other operations from the same user at the same moment, the second operation is rejected, and the error “Only one request may be made against a session at a time” is sent. The error indicates that the per user governance limit is exceeded.

Using the same login from multiple client applications, or multiple instances of the same application is NOT supported.

If you have a SuiteCloud Plus license, you can designate users as concurrent SOAP web services users. One SuiteCloud Plus license allows you to designate one user as concurrent SOAP web services user. One concurrent SOAP web services user can send up to 10 concurrent SOAP web services requests.

**Note:** If the total account limit for concurrent requests is exceeded, one concurrent SOAP web services user might not be able to send 10 concurrent requests.

**Account Limits**

Note that starting from version 2017.2, web services concurrency is not only governed separately per user and authentication method, but also per account. For information about account concurrency limits, see Web Services and RESTlet Concurrency Governance.

Each SuiteCloud Plus license increases the account concurrency governance limit by 10, meaning that you can have 10 additional parallel requests at the same per account. The account base limit depends on your service tier. For details, see Enabling Web Services Concurrent Users with SuiteCloud Plus.
Manually Managing Cookies

When submitting SOAP web services requests, the NetSuite server can accept any SOAP document as long as it is valid against the NetSuite WSDL, and it does not matter how the SOAP document was generated. You must ensure that all cookies received from the server are managed and passed back to the server in subsequent calls. There are a total of three cookies that are initially returned in the HTTP header of the login response, including the JSESSIONID.

Note: The JSESSIONID cookie is only generated if you use the Login or the ssoLogin operation for authentication. If you use request-level credentials, this cookie is not generated.

Ensure that your client persists any cookies that our server sends you (re-submits the cookie on the next request). In Axis, this is accomplished by enabling MaintainSession on the generated Binding object. In .NET, this involves allocating a CookieContainer. If persistence is not maintained, you will receive the following error message: SESSION_TIMED_OUT — “Your connection has timed out. Please log in again.”

Example

The following code shows the cookie information returned in the login response. The cookie information is contained in the Set-Cookie lines of the code.

```
HTTP/1.1 200 OK
Date: Mon, 16 Oct 2017 13:34:48 GMT
Content-Type: text/xml; charset=utf-8
X-N-OperationId: a8f84e6e-6300-4f92-9a1a-30fae3da1978
X-NetSuite-Approute: host=test.eng.netsuite.com;enter=2017-10-16
  06:34:48.716;buf=785;versrc=passport;apco=err_cxn;ver=2018.1.0;cid=000;exit=2017-10-16
  06:34:48.720;events=setcookie,rule
Set-Cookie: JSESSIONID=iR8lZOFn88GFV23ZVnnS0BUTi6V2fJZ2zi102XF9gyXWNTVrnxrXrWv0DNWY-gk5mBu075gVslo42EfwTm-
q1KnDvS5yxn(nc-Yv8oDqJVTzHokXqGvoyn0jW-747725890); path=/; HttpOnly
Set-Cookie: NS_ROUTING_VERSION=LAGGING; path=/
Set-Cookie: NS_VER=2017.2.0; domain=test.eng.netsuite.com; path=/
P3P: CP=CAO PSAa OUR BUS PUR
Vary: User-Agent
Connection: close
```

For each subsequent request, all of the cookies must be returned in the HTTP header to maintain the session as follows:

C#

```
NetSuiteService nss = new NetSuiteService();
nss.Url = https://webservices.netsuite.com/services/NetSuitePort_2017_1_0 ;
nss.CookieContainer = new CookieContainer();
```

Note: This snippet assumes that your integration includes a Using System.Net; statement.

Java with Axis 1.4

```
NetSuiteServiceLocator service = new
    DataCenterAwareNetSuiteServiceLocator(_properties.getProperty("login.acct"));
```
Encryption for SOAP Web Services

SOAP web services in NetSuite are protected by TLS encryption. Only requests sent using TLS encryption are granted access. For more information, see the help topic Supported TLS Protocol and Cipher Suites.

Custom Field Security

Custom field security can be applied on a per-field basis. If field security has been applied to a field in the UI, the custom field schema will include the field level security metadata. Consequently, users should be aware that the permissions they specify on custom fields will apply to any existing integration that is in place using endpoints 2008_1 and older. Certain field permissions such as NONE and VIEW might break the integration in an unintended way.

**Note:** See the help topic Restricting Access to Custom Fields for details on this feature.

If the custom field security feature is turned on, SOAP web services will respect the access level set on each field. For example, if you have set a field permission to NONE, and you have code that references the field, you will not be able to read or write to that field in your code. NetSuite SOAP web services will essentially ignore that field when the SOAP request is sent. Similarly, if a field permission has been set to VIEW, you will be able to read the field's value, but you will be unable to set the field's value.

SOAP web services developers should keep custom field security in mind when designing their integrations. Before beginning a project, they should work with their company's NetSuite administrator to review which fields may have custom security applied.
Platform Features

The SOAP web services platform refers to the software infrastructure used to expose NetSuite application functionality via SOAP web services. This section describes some of the core features of the SOAP web services and includes the following sections:

- **Enabling Web Services Concurrent Users with SuiteCloud Plus**: provides an overview of the Web Services Plus license type.
- **Operations and Their Internal ID Requirements**: describes IDs and how they are used.
- **Using SOAP web services to Send Data from a CSV File**: describes how to use the NetSuite Data Loader tool.
- **Searching for SOAP Web Services Log Information**: describes how to search for SOAP web services operation and record information.
- **Web Services Operations Search Type**: describes the SOAP web services operation fields available for search.
- **Web Services Record Processing Search Type**: describes the SOAP web services record fields available for search

**Important:** NetSuite hosts customer accounts in multiple data centers. For that reason, the correct URL for SOAP web services access varies depending on the data center hosting the account.

For details about dynamic discovery of URLs, see *Effects of Accounts’ Host Data Center on URLs* and *Dynamic Discovery of URLs for SOAP Web Services and RESTlet Clients*.

### Enabling Web Services Concurrent Users with SuiteCloud Plus

Your account must have the web services feature enabled before you can assign a user to be a web services concurrent user. See *Enabling the SOAP Web Services Feature* for details.

Any user can be designated as concurrent web services user if one or more SuiteCloud Plus license has been purchased.

Generally, you can purchase one SuiteCloud Plus license per NetSuite account. Contact your sales representative to discuss any need for additional licenses.

Web services and RESTlet concurrency is also governed per account. The new account governance limit applies to the combined total of web services and RESTlet requests. Due to this change, if a designated user makes 10 concurrent web services requests, the number of successful requests depends on other web services or REST requests that are being processed. Each SuiteCloud Plus license increases the account base limit for concurrent web services requests by 10. The account base limit depends on your service tier.

**Note:** The account limit is lenient, and it is possible to temporarily exceed the limit.

**Important:** If you use token-based authentication with SOAP web services, a single user can send concurrent requests up to the account limit. For more information, see *SOAP Web Services Governance for Token-Based Authentication*.

For more information about account base limits and the concurrency governance, see *Web Services and RESTlet Concurrency Governance*. 

To designate a user as a concurrent web services user:

1. Open the employee record.
2. Click the Access subtab.
3. Check the Concurrent Web Services User box.
4. Click Save.

Operations and Their Internal ID Requirements

The following table lists the ID requirements for each operation. Note that IDs are only required in calls where specific records corresponding to the call exist in the NetSuite database. For example, on an add operation, no ID is required in the request since the record does not yet exist.

An internal ID is created by the system and returned in the response. On search operations, because a specific record is not being called, an internal ID is not required in the request. However, an internal ID for each record found is returned in the response.

Note: Since external IDs are provided by the client application, if an external ID is desired for a record, it can be submitted on an Add operation.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Record ID Required in Request</th>
<th>Record ID Returned in Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>login</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>logout</td>
<td>n/a</td>
<td>n/a</td>
</tr>
<tr>
<td>add/addList</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>update/updateList</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>delete/deleteList</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>getDeleted</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>get/getList</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Operation</td>
<td>Record ID Required in Request</td>
<td>Record ID Returned in Response</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>getAll</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>search/searchNext/searchMore/searchMoreWithId</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>getSavedSearch</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The following samples show the use of internal IDs as they pertain to operations. Click to see samples for any of the following:

- add operation
- update operation
- deleteList operation
- getList operation
- search operation

### add operation

For an add operation, an internal ID is *not* required in the request since the record does not yet exist. The internal ID (in this case 100101) is returned in the response.

**Note:** There are internal IDs listed in the request, but these internal IDs are *embedded* in the request and do not correspond to the actual record being added (an event) but to other existing records associated with the event record being added.

#### Request

```xml
<soap:Body>
<platformMsgs:add>
<platformMsgs:record xsi:type="actSched:CalendarEvent">
  <actSched:title>Web Services Meeting</actSched:title>
  <actSched:organizer internalId="-5" type="calendarEvent" xsi:type="platformCore:RecordRef">
    <platformCore:name>Mr. Wolfe</platformCore:name>
  </actSched:organizer>
  <actSched:location>Main Conference Room</actSched:location>
  <actSched:attendeeList replaceAll="true" xsi:type="actSched:CalendarEventAttendeeList">
    <actSched:attendee xsi:type="actSched:CalendarEventAttendee">
      <actSched:sendEmail>false</actSched:sendEmail>
      <actSched:attendee internalId="21" type="calendarEvent" xsi:type="platformCore:RecordRef"/>
      <actSched:response>_accepted</actSched:response>
      <actSched:attendance>_optional</actSched:attendance>
    </actSched:attendee>
    <actSched:attendee xsi:type="actSched:CalendarEventAttendee">
      <actSched:sendEmail>false</actSched:sendEmail>
      <actSched:attendee internalId="27" type="calendarEvent" xsi:type="platformCore:RecordRef"/>
      <actSched:response>_accepted</actSched:response>
      <actSched:attendance>_optional</actSched:attendance>
    </actSched:attendee>
  </actSched:attendeeList>
</platformMsgs:record>
</platformMsgs:add>
</soap:Body>
```
Operations and Their Internal ID Requirements

Response

```
<soapenv:Body>

<addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <writeResponse name="urn:messages_2017_1.platform.webservices.netsuite.com">
    <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"/>
    <baseRef internalId="100101" type="calendarEvent" xsi:type="ns2:RecordRef" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
  </writeResponse>
</addResponse>

</soapenv:Body>
```

update operation

In this example the record added above is being updated — since the internal ID matches the one created in the add operation (100101). Here, the internal ID and the record type are required in the request and both are also returned in the response.

Request

```
<soap:Body>

<platformMsgs:update>
  <platformMsgs:record internalId="100101" xsi:type="actSched:CalendarEvent">
    <actSched:title>Web Services Meeting (Platform)</actSched:title>
  </platformMsgs:record>
</platformMsgs:update>

</soap:Body>
```

Response

```
<soapenv:Body>

<updateResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <writeResponse name="urn:messages_2017_1.platform.webservices.netsuite.com">
    <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"/>
    <baseRef internalId="100101" type="calendarEvent" xsi:type="ns2:RecordRef" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
  </writeResponse>
</updateResponse>

</soapenv:Body>
```

deleteList operation

In the following delete request, the internal IDs are required for the request and returned in the response. In this case, three event records are deleted (100101, 100102, and 100103).
Request

```xml
<soap:Body>
<platformMgs:deleteList>
  <platformMgs:baseRef internalId="100101" type="calendarEvent" xsi:type="platformCore:RecordRef"/>
  <platformMgs:baseRef internalId="100102" type="calendarEvent" xsi:type="platformCore:RecordRef"/>
  <platformMgs:baseRef internalId="100103" type="calendarEvent" xsi:type="platformCore:RecordRef"/>
</platformMgs:deleteList>
</soap:Body>
```

Response

```xml
<soapenv:Body>
<deleteListResponse ns xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
 <writeResponseList ns xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <writeResponse>
   <ns1:status.isSuccess="true" ns xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
    <baseRef internalId="100101" type="calendarEvent" xsi:type="ns2:RecordRef" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
   </ns1:status.isSuccess="true"/>
  </writeResponse>
  <writeResponse>
   <ns3:status.isSuccess="true" ns xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
    <baseRef internalId="100102" type="calendarEvent" xsi:type="ns4:RecordRef" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com"/>
   </ns3:status.isSuccess="true"/>
  </writeResponse>
  <writeResponse>
   <ns5:status.isSuccess="true" ns xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
    <baseRef internalId="100103" type="calendarEvent" xsi:type="ns6:RecordRef" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.com"/>
   </ns5:status.isSuccess="true"/>
  </writeResponse>
 </writeResponseList>
</deleteListResponse>
</soapenv:Body>
```

getList operation

In this example, an internal ID is called for each record to be retrieved — in this case, three different event records. Again, the internal ID is required for the request and returned in the response.

Request

```xml
<soap:Body>
<platformMgs:getList>
  <platformMgs:baseRef internalId="100104" type="calendarEvent" xsi:type="platformCore:RecordRef"/>
  <platformMgs:baseRef internalId="100105" type="calendarEvent" xsi:type="platformCore:RecordRef"/>
  <platformMgs:baseRef internalId="100106" type="calendarEvent" xsi:type="platformCore:RecordRef"/>
</platformMgs:getList>
</soap:Body>
```
Response

```xml
<soapenv:Body>
<getListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<readResponseList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<readResponse>
    <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"/>
    <record internalId="100104" xsi:type="ns2:CalendarEvent" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns2="urn:scheduling_2017_1.activities.webservices.netsuite.com">
        <ns2:title>Customization Meeting</ns2:title>
        ........[more fields]
    </record>
</readResponse>
<readResponse>
    <ns8:status isSuccess="true" xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com"/>
    <record internalId="100105" xsi:type="ns9:CalendarEvent" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns9="urn:scheduling_2017_1.activities.webservices.netsuite.com">
        <ns9:title>Web Services Meeting (Records)</ns9:title>
        ........[more fields]
    </record>
</readResponse>
<readResponse>
    <ns15:status isSuccess="true" xmlns:ns15="urn:core_2017_1.platform.webservices.netsuite.com"/>
    <record internalId="100106" xsi:type="ns16:CalendarEvent" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns16="urn:scheduling_2017_1.activities.webservices.netsuite.com">
        <ns16:title>Web Services Meeting</ns16:title>
        ........[more fields]
    </record>
</readResponse>
</readResponseList>
</getListResponse>
</soapenv:Body>
```

search operation

For the search operation, an internal ID is not required for the request but is returned for each record found that matches the specified criteria. In this case, two event records are returned for a search request calling for each event record that contains *web services* in the title field.

(Request)

```xml
<soap:Body>
<platformMgs:search>
    <platformMgs:searchRecord xsi:type="actSched:CalendarEventSearch">
        <actSched:title operator="contains" xsi:type="platformCore:SearchStringField">
            <platformCore:searchValue>web services</platformCore:searchValue>
        </actSched:title>
    </platformMgs:searchRecord>
</platformMgs:search>
</soap:Body>
```
Using SOAP web services to Send Data from a CSV File

If you have external data stored in a CSV file and want to load this data into NetSuite, you can use the new NetSuite Data Loader sample application. With this application, you can send SOAP web services requests that use the data in your CSV files. This approach is beneficial because it lets you use parallel connections, as available based on your NetSuite licensing.

**Important:** NetSuite Data Loader is intended as a demo application. No regular maintenance is provided for this application.

NetSuite Data Loader

NetSuite Data Loader is a command line Java application. It requires Gradle and Java JDK 6 to build. The source code is available for download.
To use the NetSuite Data Loader:

1. Make sure that you are logged in to NetSuite as administrator.
2. Make sure you have enabled the SOAP web services feature at Setup > Company > Setup Tasks > Enable Features, on the SuiteCloud tab.
4. Paste the URL in the address bar of your browser.
5. Substitute your account ID in the URL.
6. Press Enter.
7. Click the link for the .zip file.
8. Read and agree to the license terms.
9. Wait for download to finish.
10. Navigate to the tool's doc directory.
11. Review the README file for more detailed instructions.

Searching for SOAP Web Services Log Information

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

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

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

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

Note the following:

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

Permissions for Accessing SOAP Web Services Searches

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

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

Web Services Operations Search Type

The following SOAP web services operation fields are available for search:

- **API Version** — The API version used to perform the operation.
- **Authentication Method** — The authentication method used by the operation.
- **Job ID** — The Job ID.
- **Time** — The date and time the operation was performed.
- **Duration** — The length of time (in seconds) it took to process the operation.
- **Operation** — The SOAP web services operation.
- **Operation Status** — The operation status; failed, finished, finished with errors, processing, or rejected.
- **User Email** — The email address associated with the account that performed the operation.
- **Records In Request** — The number of records in the request.
- **Records in Response** — The number of records in the response.
- **Failed Records** — The number of records that failed.
- **Service Type** — The type of service used: SYNC or ASYNC.
- **Integration** — The related integration record.
- **Finished Records** — The number of records completed for the selected operation. This column is available for advanced searches.
- **Request** — A link to the SOAP request. This column is available for advanced searches.
- **Response** — A link to the SOAP response. This column is available for advanced searches.

The following image shows the results of a web services operations search.
Web Services Record Processing Search Type

The following SOAP web services record fields are available for search:

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

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

This section describes the following:

- **Built-in Types** — primitive or derived datatypes defined in the XML Schema specification
- **Complex Types** — NetSuite derived datatypes
- **Custom Field Types** — datatypes used for custom fields
- **Search Types** — datatypes used for search records
- **Platform Enumerations** — datatypes used to populate system defined lists

A WSDL document in SOAP web services uses the XML Schema to define the structure, semantics and constraints of the messages sent between the sender and recipient. The XML Schema is a W3C standard.

There is a strong analogy between object-oriented programming and XML Schema. A type defined in XML Schema can be used to represent an instance of that type in an XML document. Therefore, the body of a SOAP message consists of a combination of such instances of an XML Schema type.

### Built-in Types

Built-in (or primitive) types are those that are not defined in terms of other datatypes. They are used as a standardized way to define, send, receive and interpret basic data types in SOAP messages. Primitive data types used in the SOAP web services can be modified for display purposes. For example, although a price field may be passed in the SOAP messages using an integer primitive data type, the NetSuite UI may format the value with a currency symbol for display purposes.

Of the extensive set of built-in (or primitive) types provided by the XML Schema language specification, the SOAP web services implementation uses the following built-in types. For detailed information on XML Schema built-in types, refer to [http://www.w3.org/TR/xmlschema-2/](http://www.w3.org/TR/xmlschema-2/).

- **string** — represents character strings in XML
- **int** — derived from the decimal type and represents the standard concept of the integer numbers
- **double** — is patterned after the IEEE double-precision 64-bit floating point type
- **boolean** — represents the concept of binary-valued logic: \{true, false\}. In NetSuite, a boolean field can be set to true, false, 1 to indicate true, or 0 to indicate false.
  - In the 2010.2 endpoint and later, SOAP web services validates boolean values. Each value must be a true xsd boolean \(0,1,\text{true},\text{false}\) as per the W3C specification. For details, see the specification.
  - In endpoints prior to 2010.2, invalid values are treated as false.
- **dateTime** — represents integer-valued year, month, day, hour and minute properties. For example, `2005-09-21T15:24:00.000-07:00`, where 2005-09-21 is the date, 15:24:00.000 is the time and -07:00 is the timezone offset. For more information about the dateTime type, see [Datetime Fields](#).

### Complex Types

Complex types are structured types built by aggregating elements of simple or previously defined complex types. NetSuite complex types are defined in the platform core and platform core types XSDs. All NetSuite record types ultimately reference one of these complex types.

NetSuite complex types include the following:
Complex Types

- Passport
- TokenPassport
- Record
- RecordList
- BaseRef
- RecordRef
- CustomRecordRef
- ListOrRecordRef
- Status
- StatusDetail
- NullField
- ReadResponse
- ListReadResponse
- ListWriteResponse
- WriteResponse

Passport

The Passport type is used by the login operation to encapsulate the security credentials required for authentication. The passport type is defined in the core.xsd.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>email</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>password</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>account</td>
<td>xsd:string</td>
<td>Y</td>
<td>Must correspond to a valid NetSuite account number</td>
</tr>
<tr>
<td>role</td>
<td>RecordRef</td>
<td>Y</td>
<td>A role is a set of permissions that lets a user access specific areas of data. If not set, NetSuite uses the default Web Services role set for this user.</td>
</tr>
</tbody>
</table>

The default web services role for a specific user is dependent on the values set when setting the permissions for a particular role. These are set through the UI in Setup > Users/Roles > Manage Roles.

TokenPassport

The TokenPassport type is used by SOAP web services requests that use token-based authentication. The TokenPassport type is defined in the core.xsd.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>account</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>consumerKey</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>token</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>nonce</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>
### Complex Types

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>timestamp</td>
<td>xsd:long</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>signature</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** For information about the TokenPassport type, see [TokenPassport Complex Type](#).

## Record

The Record type is an abstract type used as the parameter for the add, addList, delete, deleteList, update and updateList operations. It is also returned in the get, getList, search, searchMore and searchNext operations. All business object types extend the Record type. The record type is defined in core.xsd.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xsd:string</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>xsi:type</td>
<td>xsi:type (attribute)</td>
<td>N</td>
<td>This is a field (attribute) that is automatically implemented by the XML Schema. The value should represent the concrete Record type such as Customer or Event.</td>
</tr>
<tr>
<td>nullFieldList</td>
<td>NullFields[]</td>
<td>N</td>
<td>A list of fields that are to be set to null explicitly in the update operation.</td>
</tr>
</tbody>
</table>

**Note:** You cannot set the Custom Form field to NULL. Any request to set this field to NULL is ignored.

## RecordList

The RecordList type is an array of the Record type. The recordList type is defined in core.xsd.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>Record[]</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

## BaseRef

The BaseRef type is an abstract type used to reference any existing record in NetSuite including other business records and custom records. The BaseRef type is defined in core.xsd.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>BaseRef</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

## RecordRef

The RecordRef type is used to reference an existing record in NetSuite in get operations. Typically, a RecordRef references one of the following:
Another business object such as a customer or sales order
- An entry in a user defined list such as the category list for contacts or sales tax items

The recordRef type descends from BaseRef and is defined in core.xsd.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>See Using Internal IDs, External IDs, and References.</td>
</tr>
<tr>
<td>type</td>
<td>xsd:string (attribute)</td>
<td>N</td>
<td>Reference to a value in a SOAP web services only system list. If the type is known by context, the type is NOT required.</td>
</tr>
<tr>
<td>name</td>
<td>xsd:string</td>
<td>N</td>
<td>This is a read-only field that is populated by NetSuite when it's a part of a get or search response. If this field is populated during a write operation, it will be ignored.</td>
</tr>
</tbody>
</table>

CustomRecordRef

The CustomRecordRef type is used to reference any existing custom record in NetSuite. The CustomRecordRef type descends from BaseRef and is defined in core.xsd.

⚠️ **Important:** Setting the RecordRef.type to `customRecord` on an add does NOT return a CustomRecord. You must use CustomRecordRef. The CustomRecordRef has a typeId to indicate which kind of CustomRecord it is.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>References the primary record internal Id. This Id corresponds to the type of custom record. For a list of possible values, see ListOrRecordRef.</td>
</tr>
<tr>
<td>typeId</td>
<td>xsd:string</td>
<td>Y</td>
<td>References the custom record type Id.</td>
</tr>
<tr>
<td>type</td>
<td>xsd:string (attribute)</td>
<td>N</td>
<td>Reference to a value in a SOAP web services only system list.</td>
</tr>
<tr>
<td>name</td>
<td>xsd:string</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

ListOrRecordRef

The listOrRecordRef type is defined in core.xsd.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>See Using Internal IDs, External IDs, and References.</td>
</tr>
<tr>
<td>externalId</td>
<td>xsd:string (attribute)</td>
<td>N</td>
<td>Use to reference records by their external ID in select and multi-select custom fields.</td>
</tr>
<tr>
<td>typeId</td>
<td>xsd:string (attribute)</td>
<td>N</td>
<td>Reference to a value in a SOAP web services only system list.</td>
</tr>
<tr>
<td>name</td>
<td>xsd:string</td>
<td>N</td>
<td>This is a read-only field that is populated by NetSuite when it's a part of a get or search response. If this field is populated during a write operation, it is ignored.</td>
</tr>
</tbody>
</table>
Each record type in NetSuite has a corresponding internal ID (or typeId). This internal ID is required when using ListOrRecordRef since the type of record being referenced needs to be specified.

Note: To update a ListOrRecordRef object, you must use either the internal ID or the external ID in your request. Using the name field only does not update the object.

For example, in the following code a new ListOrRecordRef object is created. The list references a specific Entity record as designated by the internalId of 1011 and specifies that the record is of the type customer (-2). Note that customer records have an internal ID of -2 as shown in the table below.

```java
ListOrRecordRef[] fieldNameEntity = new ListOrRecordRef[1];
fieldNameEntity[0] = new ListOrRecordRef();
fieldNameEntity[0].setInternalId("1011");
fieldNameEntity[0].setTypeId("-2");
```

Status

The Status type contains an array of objects of type StatusDetail. The status type is defined in core.xsd.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>statusDetail</td>
<td>StatusDetail []</td>
<td>N</td>
<td>Used to capture the specific details for the status. See StatusDetail.</td>
</tr>
<tr>
<td>isSuccesss</td>
<td>xsd:Boolean (attribute)</td>
<td>Y</td>
<td>Indicates whether the status is successful or not. If false, one or more statusDetail objects are populated.</td>
</tr>
</tbody>
</table>

StatusDetail

The StatusDetail type is used to capture the specific details for the status. The statusDetail type is defined in core.xsd.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>code</td>
<td>xsd:string</td>
<td>Y</td>
<td>The status code. See SOAP Web Services Error Handling and Error Codes for a listing of codes.</td>
</tr>
<tr>
<td>message</td>
<td>xsd:string</td>
<td>Y</td>
<td>The detailed message for this status. See SOAP Web Services Error Handling and Error Codes for details.</td>
</tr>
<tr>
<td>type</td>
<td>xsd:string</td>
<td></td>
<td>Reference to a value in a SOAP web services only system list. Values: error, warning See SOAP Web Services Error Handling and Error Codes for details.</td>
</tr>
</tbody>
</table>

NullField

The NullField type is defined in core.xsd. It contains the following fields.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>xsd:string</td>
<td>Y</td>
<td>Name of the field to be null. The specified name must exactly match an existing field name.</td>
</tr>
</tbody>
</table>
Note: You cannot set the Custom Form field to NULL. Any request to set this field to NULL is ignored.

ReadResponse

The ReadResponse type is used by the following read operations.

- The getResponse output message for the get operation.
- The searchResponse output message for the search operations.

These types have a field named readResponse of the type ReadResponse. The ReadResponse type is defined in message.xsd.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>recordRef</td>
<td>RecordRef</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

ListReadResponse

The ListReadResponse type is used by the following operations.

- The GetListResponse output message for the getList operation.

These types have a field named response of type ListReadResponse. The ListReadResponse type is defined in message.xsd.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>ReadResponse[]</td>
<td>Y</td>
<td>An array of ReadResponse types.</td>
</tr>
</tbody>
</table>

ListWriteResponse

The ListWriteResponse type is used by the following operations.

- The AddListResponse output message for the addList operation.
- The UpdateListResponse output message for the updateList operation.
- The DeleteListResponse output message for the deleteList operation.

These types have a field named response of type ListWriteResponse. The ListWriteResponse type is defined in message.xsd.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>WriteResponse[]</td>
<td>Y</td>
<td>An array of WriteResponse types.</td>
</tr>
</tbody>
</table>

WriteResponse

The WriteResponse type is used by the following operations:
The AddResponse output message for the add operation
The UpdateResponse output message for the update operation
The DeleteResponse output message for the delete operation

These types have a field named writeResponse of type WriteResponse. The WriteResponse type is defined in message.xsd.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>recordRef</td>
<td>RecordRef</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

**Custom Field Types**

Custom fields are represented by the type CustomFieldRef, which is an abstract type. The table below contains a list of concrete custom field types that extend the CustomFieldRef type. Each type is followed by its corresponding type in the UI.

<table>
<thead>
<tr>
<th>XML Schema Type</th>
<th>Custom Field Type in UI</th>
</tr>
</thead>
<tbody>
<tr>
<td>LongCustomFieldRef</td>
<td>Integer</td>
</tr>
<tr>
<td>DoubleCustomFieldRef</td>
<td>Decimal Number</td>
</tr>
<tr>
<td>BooleanCustomFieldRef</td>
<td>Check Box</td>
</tr>
<tr>
<td>StringCustomFieldRef</td>
<td>Free-Form Text, Text Area, Phone Number, E-mail Address, Hyperlink, Rich Text</td>
</tr>
<tr>
<td>DateCustomFieldRef</td>
<td>Date, Time of Day, or Date/Time (both in one field)</td>
</tr>
<tr>
<td>SelectCustomFieldRef</td>
<td>List/Record Document</td>
</tr>
<tr>
<td>MultiSelectCustomFieldRef</td>
<td>Multiple Select</td>
</tr>
</tbody>
</table>

**CustomFieldRef**

The CustomFieldRef type is an abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>References a unique instance of a custom field type.</td>
</tr>
</tbody>
</table>

To locate the internal ID for a specific custom field in the UI, go to Customization > Lists, Records, & Fields > [Custom Field] (where [Custom Field] is the type of custom field such as CRM). The internal IDs for each custom field that has been created is listed in the ID column in the UI.
LongCustomFieldRef

The LongCustomFieldRef type extends the CustomFieldRef abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>xsd:int</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>References a unique instance of a custom field type.</td>
</tr>
</tbody>
</table>

DoubleCustomFieldRef

The DoubleCustomFieldRef type extends the CustomFieldRef abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>xsd:double</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>References a unique instance of a custom field type.</td>
</tr>
</tbody>
</table>

BooleanCustomFieldRef

The BooleanCustomFieldRef type extends the CustomFieldRef abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>xsd:boolean</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>References a unique instance of a custom field type.</td>
</tr>
</tbody>
</table>

StringCustomFieldRef

The StringCustomFieldRef type extends the CustomFieldRef abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>References a unique instance of a custom field type.</td>
</tr>
</tbody>
</table>

DateCustomFieldRef

The DateCustomFieldRef type extends the CustomFieldRef abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>xsd:datetime</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>References a unique instance of a custom field type.</td>
</tr>
</tbody>
</table>
**SelectCustomFieldRef**

The SelectCustomFieldRef type extends the CustomFieldRef abstract type. This references a single ListOrRecordRef and also requires an InternalId attribute to indicate the field name.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>ListOrRecordRef</td>
<td>Y</td>
<td>A single ListOrRecordRef.</td>
</tr>
<tr>
<td>internalId</td>
<td>xsd:string</td>
<td>Y</td>
<td>References a unique instance of a custom field type.</td>
</tr>
</tbody>
</table>

**MultiSelectCustomFieldRef**

The MultiSelectCustomFieldRef type extends the CustomFieldRef abstract type. This references an array of ListOrRecordRef's and also requires an internalId attribute to indicate the field name.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>ListOrRecordRef</td>
<td>Y</td>
<td>An array of type RecordRef</td>
</tr>
<tr>
<td>internalId</td>
<td>xsd:string</td>
<td>Y</td>
<td>References a unique instance of a custom field type.</td>
</tr>
</tbody>
</table>

**CustomFieldList**

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>customFieldRef</td>
<td>Y</td>
<td>An array of type customFieldRef. The actual entries in the array will be of a concrete type that extends customFieldRef.</td>
</tr>
</tbody>
</table>

The following is an XML excerpt from a SOAP body that illustrates a custom field list that contains all the available custom field types.

**SOAP Sample**

```xml
<listRel:customFieldList xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
  <platformCore:customField internalId="526" scriptId="custentity_exchange_rate" xsi:type="platformCore:DoubleCustomFieldRef">
    <platformCore:value>2.35</platformCore:value>
  </platformCore:customField>
  <platformCore:customField internalId="524" scriptId="custentity_soft_skills" xsi:type="platformCore:MultiSelectCustomFieldRef">
    <platformCore:value internalId="1">
      <platformCore:name>Communication</platformCore:name>
    </platformCore:value>
    <platformCore:value internalId="5">
      <platformCore:name>Teamwork</platformCore:name>
    </platformCore:value>
  </platformCore:customField>
  <platformCore:customField internalId="146" scriptId="custentity_checkbox" xsi:type="platformCore:BooleanCustomFieldRef">
```

---

SuiteTalk SOAP Web Services Platform Guide

Oracle NetSuite
Search Types

Every NetSuite record that supports search has corresponding search and advanced search objects. For example, the SOAP listRel XSD contains a Customer object, as well as its corresponding CustomerSearch, CustomerSearchAdvanced, and CustomerSearchRow search objects.

- When using a `<Record>` Search object, all search fields within this search object belong to one of the types described in Search XML Schema Types.
- When searching on custom records or custom fields, all search fields belong to one of the types described in Search Custom Field XML Schema Types.
- When using either the `<Record>` SearchAdvanced or `<Record>` SearchRow search objects, all search fields within these “advanced search” objects belong to one of the types described in Search Column Custom XML Schema Types or Search Column Custom XML Schema Types.

Search XML Schema Types

The following sections define available search types. Every search field within a search object type belongs to one of these search types.

### SearchPreferences

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
</tr>
</thead>
<tbody>
<tr>
<td>bodyFieldsOnly</td>
<td>boolean</td>
<td>Y/N</td>
</tr>
<tr>
<td>pageSize</td>
<td>int</td>
<td>Y/N</td>
</tr>
<tr>
<td>returnSearchColumns</td>
<td>boolean</td>
<td>Y/N</td>
</tr>
</tbody>
</table>
## SearchRequest

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>preferences</td>
<td>boolean</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>searchRecord</td>
<td>SearchRecord</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

## SearchResult

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>totalRecords</td>
<td>int</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>pageSize</td>
<td>int</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>totalPages</td>
<td>int</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>pageIndex</td>
<td>int</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>searchId</td>
<td>string</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td>status</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>recordList</td>
<td>platformCore:RecordList</td>
<td></td>
<td></td>
</tr>
<tr>
<td>searchRowList</td>
<td>platformCore:SearchRowList</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## SearchStringField

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator</td>
<td>platformCoreTyp: SearchStringFieldOperator (attribute)</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>searchValue</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

**Notes:** Reference to a value in a system list. For more information on available values, see Platform Enumerations.

## SearchBooleanField

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator</td>
<td>xsd:boolean</td>
<td>Y</td>
<td>The available values are true or false.</td>
</tr>
</tbody>
</table>

## SearchDoubleField

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator</td>
<td>platformCoreTyp: SearchDoubleFieldOperator (attribute)</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>searchValue</td>
<td>xsd:double</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

**Notes:** Reference to a value in a system list. For more information on available values, see Platform Enumerations.
### Search Types

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue2</td>
<td>xsd:double</td>
<td>N</td>
<td>If the operator is between or notBetween searchValue2 must be populated.</td>
</tr>
</tbody>
</table>

#### SearchLongField

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator</td>
<td>platformCoreTyp: SearchLongFieldOperator(attribute)</td>
<td>Y</td>
<td>Reference to a value in a system list. For more information on available values, see Platform Enumerations.</td>
</tr>
<tr>
<td>searchValue</td>
<td>xsd:long</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>searchValue2</td>
<td>xsd:long</td>
<td>N</td>
<td>If the operator is between or notBetween searchValue2 must be populated.</td>
</tr>
</tbody>
</table>

#### SearchTextNumberField

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator</td>
<td>platformCoreTyp: SearchTextNumberFieldOperator(attribute)</td>
<td>Y</td>
<td>Reference to a value in a system list. For more information on available values, see Platform Enumerations.</td>
</tr>
<tr>
<td>searchValue</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>searchValue2</td>
<td>xsd:string</td>
<td>N</td>
<td>If the operator is between or notBetween searchValue2 must be populated.</td>
</tr>
</tbody>
</table>

#### SearchDateField

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator</td>
<td>platformCoreTyp: SearchDate(attribute)</td>
<td>Y</td>
<td>Reference to a value in a system list. For more information on available values, see Platform Enumerations.</td>
</tr>
<tr>
<td>predefinedSearchValue</td>
<td>platformCoreTyp: SearchDate</td>
<td>N</td>
<td>Reference to a value in a system list. For more information on available values, see Platform Enumerations.</td>
</tr>
<tr>
<td>searchValue</td>
<td>xsd:dateTime</td>
<td>N</td>
<td>Either predefinedSearchValue or searchValue should be populated.</td>
</tr>
<tr>
<td>searchValue2</td>
<td>xsd:dateTime</td>
<td>N</td>
<td>If the operator is between or notBetween searchValue2 must be populated.</td>
</tr>
</tbody>
</table>

#### SearchMultiSelectField

This search type is used to specify a list of one or more internal IDs that reference other user defined records in the system.
Search Types

**Note:** Note that the maximum number of values that can be specified in a MultiSelectField is 1000.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator</td>
<td>platformCoreTyp: SearchMultiSelectFieldOperator</td>
<td>Y</td>
<td>Reference to a value in a system list. For more information on available values, see Platform Enumerations.</td>
</tr>
<tr>
<td>searchValue</td>
<td>platformCore: RecordRef</td>
<td>Y</td>
<td>An array of type SearchMultiSelectRefValue.</td>
</tr>
</tbody>
</table>

**SearchEnumMultiSelectField**

This search type is used to specify a list of one or more system defined constants.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>operator</td>
<td>platformCoreTyp: SearchEnumMultiSelectFieldOperator</td>
<td>Y</td>
<td>Reference to a value in a system list. For more information on available values, see Platform Enumerations.</td>
</tr>
<tr>
<td>searchValue</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

**Search Custom Field XML Schema Types**

The following sections define the available search types for custom fields. Every search field within a search object type belongs to one of these search types.

**SearchCustomField**

This is an abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>References a unique instance of a custom field</td>
</tr>
</tbody>
</table>

**SearchStringCustomField**

The SearchStringCustomField type extends the SearchCustomField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>Inherited from the SearchCustomField. Reference a unique instance of a custom field.</td>
</tr>
<tr>
<td>operator</td>
<td>platformCoreTyp: SearchStringFieldOperator (attribute)</td>
<td>Y</td>
<td>The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see Platform Enumerations.</td>
</tr>
<tr>
<td>searchValue</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

**SearchBooleanCustomField**

The SearchBooleanCustomField type extends the SearchCustomField abstract type.
### SearchTypes

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>Inherited from the SearchCustomField. Reference a unique instance of a custom field.</td>
</tr>
<tr>
<td>searchValue</td>
<td>xsd: boolean</td>
<td>Y</td>
<td>The type is an enumeration type that restricts the value to true or false.</td>
</tr>
</tbody>
</table>

#### SearchLongCustomField

The SearchLongCustomField type extends the SearchCustomField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>Inherited from the SearchCustomField. Reference a unique instance of a custom field.</td>
</tr>
<tr>
<td>operator</td>
<td>platformCoreTyp: SearchLongFieldOperator (attribute)</td>
<td>Y</td>
<td>The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see Platform Enumerations.</td>
</tr>
<tr>
<td>searchValue</td>
<td>xsd:long</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>searchValue2</td>
<td>xsd:long</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

#### SearchDoubleCustomField

The SearchDoubleCustomField type extends the SearchCustomField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>Inherited from the SearchCustomField. Reference a unique instance of a custom field.</td>
</tr>
<tr>
<td>operator</td>
<td>platformCoreTyp: SearchDoubleFieldOperator (attribute)</td>
<td>Y</td>
<td>The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see Platform Enumerations.</td>
</tr>
<tr>
<td>searchValue</td>
<td>xsd:double</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>searchValue2</td>
<td>xsd:double</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

#### SearchDateCustomField

The SearchDateCustomField type extends the SearchCustomField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>Inherited from the SearchCustomField. Reference a unique instance of a custom field.</td>
</tr>
<tr>
<td>operator</td>
<td>platformCoreTyp: SearchDateFieldOperator (attribute)</td>
<td>Y</td>
<td>The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see Platform Enumerations.</td>
</tr>
<tr>
<td>predefinedSearchValue</td>
<td>platformCoreTyp: SearchDate</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

SuiteTalk SOAP Web Services Platform Guide
### SearchTypes

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>xsd:dateTime</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>searchValue2</td>
<td>xsd:dateTime</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

#### SearchMultiSelectCustomField

The SearchMultiSelectCustomField type extends the SearchCustomField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>Inherited from the SearchCustomField. Reference a unique instance of a custom field.</td>
</tr>
<tr>
<td>operator</td>
<td>platformCoreTyp: SearchMultiSelectFieldOperator (attribute)</td>
<td>Y</td>
<td>The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see Platform Enumerations.</td>
</tr>
<tr>
<td>searchValue</td>
<td>ListOrRecordRef</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

#### SearchEnumMultiSelectCustomField

The SearchEnumMultiSelectCustomField type extends the SearchCustomField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td>Inherited from the SearchCustomField. Reference a unique instance of a custom field.</td>
</tr>
<tr>
<td>operator</td>
<td>platformCoreTyp: SearchEnumMultiSelectFieldOperator (attribute)</td>
<td>Y</td>
<td>The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see Platform Enumerations.</td>
</tr>
<tr>
<td>searchValue</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

#### SearchCustomFieldList

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>customField</td>
<td>platformCore: SearchCustomField []</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

#### Search Column XML Schema Types

The following search types support advanced search functionality in SOAP web services.

#### SearchColumnField

This is an abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>customLabel</td>
<td>xsd:string</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SearchColumnBooleanField
The SearchColumnBooleanField type extends the SearchColumnField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>xsd:boolean</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

SearchColumnStringField
The SearchColumnStringField type extends the SearchColumnField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

SearchColumnLongField
The SearchColumnLongField type extends the SearchColumnField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>xsd:long</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

SearchColumnTextNumberField
The SearchColumnStringField type extends the SearchColumnField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

SearchColumnDoubleField
The SearchColumnDoubleField type extends the SearchColumnField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>xsd:double</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

SearchColumnDateField
The SearchColumnDateField type extends the SearchColumnField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>xsd:dateTime</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

SearchColumnEnumSelectField
The SearchColumnEnumSelectField type extends the SearchColumnField abstract type.
Search Types

### SearchColumnSelectField

The SearchColumnSelectField type extends the SearchColumnField abstract type. This references a single ListOrRecordRef and also requires an internalId attribute to indicate the field name.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>RecordRef</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

### Search Column Custom XML Schema Types

The following search types support advanced search functionality in SOAP web services.

#### SearchColumnCustomField

This is an abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>customLabel</td>
<td>xsd:string</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### SearchColumnBooleanCustomField

The SearchColumnBooleanField type extends the SearchColumnCustomField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>xsd:boolean</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

#### SearchColumnStringCustomField

The SearchColumnStringCustomField type extends the SearchColumnCustomField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

#### SearchColumnLongCustomField

The SearchColumnLongCustomField type extends the SearchColumnCustomField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>xsd:long</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>
### SearchTypes

#### SearchColumnDoubleCustomField

The `SearchColumnDoubleCustomField` type extends the `SearchColumnCustomField` abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>xsd:double</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

#### SearchColumnDateCustomField

The `SearchColumnDateCustomField` type extends the `SearchColumnCustomField` abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>xsd:dateTime</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

#### SearchColumnEnumMultiSelectCustomField

The `SearchColumnEnumMultiSelectCustomField` type extends the `SearchColumnCustomField` abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>xsd:string</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

#### SearchColumnSelectCustomField

The `SearchColumnSelectCustomField` type extends the `SearchColumnCustomField` abstract type. This references a single `ListOrRecordRef` and also requires an `internalId` attribute to indicate the field name.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>ListOrRecordRef</td>
<td>Y</td>
<td>A single ListOrRecordRef</td>
</tr>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

#### SearchColumnMultiSelectCustomField

The `SearchColumnMultiSelectCustomField` type extends the `SearchColumnCustomField` abstract type. This references an array of `ListOrRecordRef`es.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchValue</td>
<td>ListOrRecordRef[]</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>
Search Types

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalId</td>
<td>xsd:string (attribute)</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>

SearchColumnCustomFieldList

The SearchColumnCustomFieldList type extends the SearchColumnCustomField abstract type.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>XML Schema Type</th>
<th>Req</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>customField</td>
<td>SearchColumnCustomField</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sample Code

**SOAP Request — Opportunity Search**

Following is an example that contains an excerpt of the SOAP body that illustrates an opportunity search containing several search field types.

```xml
<opportunitySearch>
  <projectedTotal operator="lessThan">
    <searchValue>100000</searchValue>
  </projectedTotal>
  <title operator="contains">
    <searchValue>Enterprise</searchValue>
  </title>
  <createdDateRange operator="between">
    <fromValue>2003-10-02</fromValue>
    <toValue>2003-10-12</toValue>
  </createdDateRange>
  <opportunityStatusList operator="anyOf">
    <searchValue>inProgress</searchValue>
    <searchValue>closedWon</searchValue>
  </opportunityStatusList>
  <customFieldList>
    <customField xsi:type="SearchSelectCustomField" internalId="35" scriptId="cust_SalesEngineer" operator="equals">
      <searchValue>Buddy Williams</searchValue>
    </customField>
    <customField xsi:type="SearchBooleanCustomField" internalId="165" scriptId="cust_hasSalesEngineer" operator="true"/>
    <customField xsi:type="SearchStringCustomField" internalId="322" scriptId="cust_DemoNotes" operator="startsWith">
      <searchValue>CRM</searchValue>
    </customField>
    <customField xsi:type="SearchMultiSelectCustomField" internalId="155" scriptId="cust_ProductAreas" operator="noneOf">
      <searchValue>Inventory</searchValue>
      <searchValue>Warehousing</searchValue>
    </customField>
  </customFieldList>
</opportunitySearch>
```
# Platform Enumerations

The following search types are used throughout SOAP web services to populate system defined lists. The tables below outline the available values that should be used to populate these fields in your SOAP web services requests. These enumerations are defined in the `platformCoreTyp XSD`.

## Search Types (Table 1)

<table>
<thead>
<tr>
<th>Enumerations</th>
<th>SearchString FieldOperator</th>
<th>SearchLongField Operator</th>
<th>SearchDoubleField Operator</th>
<th>SearchPeriod Field</th>
<th>SearchTime FieldOperator</th>
</tr>
</thead>
<tbody>
<tr>
<td>is</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>isNot</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>startsWith</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doesNotStartWith</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>contains</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>doesNotContain</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>equalTo</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lessThan</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>greaterThan</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lessThanOrEqualTo</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>greaterThanOrEqualTo</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notEqualTo</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notLessThan</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notGreaterThan</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notLessThanOrEqualTo</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notGreaterThanOrEqualTo</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>between</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notBetween</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>empty</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notEmpty</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Search Types (Table 2)

<table>
<thead>
<tr>
<th>Enumerations</th>
<th>SearchDate FieldOperator</th>
<th>SearchEnum MultiSelect FieldOperator</th>
<th>SearchMultiSelectFieldOperator</th>
<th>SearchDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>anyOf</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>noneOf</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>on</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>before</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>after</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>onOrBefore</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>onOrAfter</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>within</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>empty</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enumerations</td>
<td>SearchDate FieldOperator</td>
<td>SearchEnum MultiSelect FieldOperator</td>
<td>SearchMulti SelectField Operator</td>
<td>SearchDate</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>--------------------------</td>
<td>-------------------------------------</td>
<td>----------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>notOn</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notBefore</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notAfter</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notOnOrBefore</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notOnOrAfter</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notWithin</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notEmpty</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fiscalHalfBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fiscalHalfBeforeLastToDate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fiscalQuarterBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fiscalQuarterBeforeLastToDate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fiscalYearBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fiscalYearBeforeLastToDate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fiveDaysAgo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fiveDaysFromNow</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fourDaysAgo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fourDaysFromNow</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fourWeeksStartingThisWeek</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastBusinessWeek</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastFiscalHalf</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastFiscalHalfOneFiscalYearAgo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastFiscalHalfToDate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastFiscalQuarter</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastFiscalQuarterOneFiscalYearAgo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastFiscalQuarterToDate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastFiscalQuarterTwoFiscalYearsAgo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastFiscalYearToDate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastMonth</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastMonthOneFiscalQuarterAgo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastMonthOneFiscalYearAgo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastMonthToDate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastMonthTwoFiscalQuartersAgo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastMonthTwoFiscalYearsAgo</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastRollingHalf</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastRollingQuarter</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lastRollingYear</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Search Types (Table 2)

<table>
<thead>
<tr>
<th>Enumerations</th>
<th>SearchDate FieldOperator</th>
<th>SearchEnum MultiSelect FieldOperator</th>
<th>SearchMultiSelectFieldOperator</th>
<th>SearchDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>lastWeek</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>lastWeekToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>monthAfterNext</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>monthAfterNextToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>monthBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>monthBeforeLastToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>nextBusinessWeek</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>nextFiscalHalf</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>nextFiscalQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>nextFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>nextFourWeeks</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>nextMonth</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>nextOneHalf</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>nextOneMonth</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>nextOneQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>nextOneWeek</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>nextOneYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>nextWeek</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ninetyDaysAgo</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ninetyDaysFromNow</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>oneYearBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousFiscalQuartersLastFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousFiscalQuartersThisFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousMonthsLastFiscalHalf</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousMonthsLastFiscalQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousMonthsLastFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousMonthsSameFiscalHalfLastFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousMonthsSameFiscalQuarterLastFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousMonthsThisFiscalHalf</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousMonthsThisFiscalQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousMonthsThisFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousOneDay</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousOneHalf</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousOneMonth</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousOneQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousOneWeek</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousOneYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Enumerations</td>
<td>SearchDate FieldOperator</td>
<td>SearchEnum MultiSelect FieldOperator</td>
<td>SearchMulti SelectField Operator</td>
<td>SearchDate</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>--------------------------</td>
<td>--------------------------------------</td>
<td>----------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>previousRollingHalf</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousRollingQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>previousRollingYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameDayFiscalQuarterBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameDayFiscalYearBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameDayLastFiscalQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameDayLastFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameDayLastWeek</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameDayMonthBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameDayWeekBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameFiscalHalfLastFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameFiscalHalfLastFiscalYearToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameFiscalQuarterFiscalYearBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameFiscalQuarterLastFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameMonthFiscalQuarterBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameMonthFiscalYearBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameMonthLastFiscalQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameMonthLastFiscalQuarterToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameMonthLastFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameMonthLastFiscalYearToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameWeekFiscalYearBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sameWeekLastFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sixtyDaysAgo</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>sixtyDaysFromNow</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfFiscalHalfBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfFiscalQuarterBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfFiscalYearBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfLastBusinessWeek</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfLastFiscalHalf</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfLastFiscalHalfOneFiscalYearAgo</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfLastFiscalQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfLastFiscalQuarterOneFiscalYearAgo</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfLastFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfLastMonth</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfLastMonthOneFiscalQuarterAgo</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
## Search Types (Table 2)

<table>
<thead>
<tr>
<th>Enumerations</th>
<th>SearchDate FieldOperator</th>
<th>SearchEnum MultiSelect FieldOperator</th>
<th>SearchMultiSelectFieldOperator</th>
<th>SearchDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>startOfMonthOneFiscalYearAgo</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfMonthRollingHalf</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfMonthRollingQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfMonthRollingYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfWeek</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfMonthBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfNextBusinessWeek</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfNextFiscalHalf</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfNextFiscalQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfNextFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfNextMonth</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfNextWeek</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfPreviousRollingHalf</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfPreviousRollingQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfPreviousRollingYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfSameFiscalHalfLastFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfSameFiscalQuarterLastFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfSameMonthLastFiscalQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfSameMonthLastFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfThisBusinessWeek</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfThisFiscalHalf</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfThisFiscalQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfThisFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfThisMonth</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfThisWeek</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfThisYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>startOfWeekBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>tenDaysAgo</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>tenDaysFromNow</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thirtyDaysAgo</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thirtyDaysFromNow</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thisBusinessWeek</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thisFiscalHalf</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thisFiscalHalfToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thisFiscalQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thisFiscalQuarterToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thisFiscalYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
### Platform Enumerations

#### Search Types (Table 2)

<table>
<thead>
<tr>
<th>Enumerations</th>
<th>SearchDate FieldOperator</th>
<th>SearchEnum MultiSelect FieldOperator</th>
<th>SearchMulti SelectField Operator</th>
<th>SearchDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>thisFiscalYearToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thisMonth</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thisMonthToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thisRollingHalf</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thisRollingQuarter</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thisRollingYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thisWeek</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thisWeekToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>thisYear</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>threeDaysAgo</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>threeDaysFromNow</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>threeFiscalQuartersAgo</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>threeFiscalQuartersAgoToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>threeFiscalYearsAgo</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>threeFiscalYearsAgoToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>threeMonthsAgo</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>threeMonthsAgoToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>today</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>tomorrow</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>twoDaysAgo</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>twoDaysFromNow</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>weekAfterNext</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>weekAfterNextToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>weekBeforeLast</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>weekBeforeLastToDate</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>yesterday</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

**Important:** The onOrAfter, onOrBefore, notOnOrAfter and notOnOrBefore operators only consider the date you specify in your request, and ignore the time you specify. If you need to specify not only the date but also the time in your search, use the after or before operators. Note, however, when you use these operators, the exact time specified in your request is not part of the result set.
SOAP Web Services Operations

SOAP web services exposes NetSuite as a data source for programmatic access, hence most of the data operations developers expect, such as insert, update, delete, and select/search are supported. There are also a number of operations that are available as supporting operations for data read/write operations (for example: initialize and getSelectValue), or for providing metadata (for example: getCustomizationId), or for exposing an application function to programmatic access (for example: attach, detach).

**Note:** If an operation you are looking for doesn't appear in the list of operations below, look at the Support for Existing WSDL Versions for your version of the WSDL.

See these sections for complete details on all SOAP web services operations:

- SOAP Web Services Standard Operations
- SOAP Web Services List Operations
- SOAP Web Services Asynchronous Operations

**Important:** NetSuite hosts customer accounts in multiple data centers. For that reason, the correct URL for SOAP web services access varies depending on the data center hosting the account.

For details about dynamic discovery of URLs, see Effects of Accounts' Host Data Center on URLs and Dynamic Discovery of URLs for SOAP Web Services and RESTlet Clients.

SOAP Web Services Standard Operations

The following operations are supported in SOAP web services. They are organized in alphabetical order:

<table>
<thead>
<tr>
<th>Operation / API</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>add / addList</td>
<td>Use to add one or more records into the system. The system returns a NetSuite identifier (internalId) that is unique for each record created within a record type.</td>
</tr>
<tr>
<td>attach / detach</td>
<td>Use to attach or detach another record or file to/from another record.</td>
</tr>
<tr>
<td>changeEmail</td>
<td>Use to change the email address for the account.</td>
</tr>
<tr>
<td>changePassword</td>
<td>Use to change the password for the account.</td>
</tr>
<tr>
<td>changePasswordOrEmail</td>
<td>Use to change a users email or password.</td>
</tr>
<tr>
<td>checkAsyncStatus</td>
<td>Use to check the status of an asynchronous SOAP web services submission.</td>
</tr>
<tr>
<td>delete / deleteList</td>
<td>Use to delete one or more records in the system. The records to be deleted are identified by either the internal or external ID and the record type.</td>
</tr>
<tr>
<td>get / getList</td>
<td>Use to query the system for one or more records. You must provide either the internal or external ID and the record type for each query item.</td>
</tr>
<tr>
<td>getAll</td>
<td>Use to return a list of records that do not have a search interface.</td>
</tr>
<tr>
<td>getAsyncResult</td>
<td>Use to retrieve the results of an asynchronous SOAP web services submission.</td>
</tr>
<tr>
<td>getBudgetExchangeRate</td>
<td>Use to get and filter all data related to the Budget Exchange Rates table.</td>
</tr>
<tr>
<td>Operation / API</td>
<td>Summary</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| getConsolidatedExchangeRate     | Use to get and filter all data related to the Consolidated Exchange Rates table.  
**Important:** The exposure of the consolidated exchange rate record prompted the removal of the getConsolidatedExchangeRate operation from the 2017.1 endpoint. You can still use the getConsolidatedExchangeRate operation on earlier endpoints. For more information about working with consolidated exchange rates on endpoints starting from 2017.1, see the help topic Consolidated Exchange Rate. |
| getCurrencyRate                 | Use to get currency exchange rates from the Currency Exchange Rate table.                                                                                                                                                  |
| getCustomizationId              | Use to retrieve the internalIds, externalIds, and/or scriptIds of all custom objects of a specified type.                                                                                                               |
| getDataCenterUrls               | Use for dynamic discovery of the correct URL for SOAP web services requests.  
**Important:** All SOAP web services integrations must include logic to dynamically discover the correct URL. If the getDataCenterURLs operation is not included in the endpoint you are using, use the REST roles service to obtain the correct URL. For details on this service, see The REST roles Service.  
**Important:** As of the 2019.1 SOAP web services endpoint, you must use account-specific domains with SOAP web services. You can use the SOAP getDataSetCoreUrls operation to obtain the correct domain. Or, go to Setup > Company > Setup Tasks > Company Information (Administrator) in the NetSuite UI. Your domains are listed on the Company URLs subtab. |
<p>| getDeleted                      | Use to retrieve a list of deleted records of a particular type during a specified period.                                                                                                                                   |
| getItemAvailability             | Use to retrieve the inventory availability for a specific list of items.                                                                                                                                                   |
| getPostingTransactionSummary    | Use to retrieve a summary of the actual data in an account.                                                                                                                                                    |
| getSavedSearch                  | Use to retrieve a list of existing saved searches for each record type. Specify the search record type to get a list of record references of the saved search.                                                        |
| getSelectValue                  | Use to retrieve valid values for a particular recordRef field where the referenced record type is not yet exposed in the SOAP web services API or when the logged in role does not have permission to the instances of the record type. |
| getServerTime                   | Use to get server time, resulting in more accurate and reliable syncing of data than using using local client time. The client will use the time from server to determine if the record has changed since the last synchronization. |
| initialize / initializeList     | Use to emulate the UI workflow by pre-populating fields on transaction line items with values from a related record. The initializeList operation can be used to run batch processes to retrieve initialized records. |
| login                           | Use to login into NetSuite. This operation is similar to the NetSuite UI and requires you to provide a valid username, password, role, and account number.                                                                 |</p>
<table>
<thead>
<tr>
<th>Operation / API</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>logout</td>
<td>Use to logout from the system. The logout operation invalidates the current session.</td>
</tr>
<tr>
<td>mapSso</td>
<td>Use to automate the mapping between external applications credentials and NetSuite credentials for a user.</td>
</tr>
<tr>
<td>search</td>
<td>Use to search for a set of records based on specific search criteria. This operation supports pagination, so that large result sets can be retrieved in smaller sets. For more information on how to control pagination, refer to SOAP Web Services Preferences.</td>
</tr>
<tr>
<td>searchMore</td>
<td>Used to retrieve more records after an initial search operation is invoked.</td>
</tr>
<tr>
<td>searchMoreWithId</td>
<td>Use to retrieve search results for users who send requests to NetSuite by providing request-level credentials, rather than by invoking login. In this case, users cannot call searchMore or searchNext.</td>
</tr>
<tr>
<td>searchNext</td>
<td>Use to retrieve the next set of records after an initial search operation is invoked.</td>
</tr>
<tr>
<td>ssoLogin</td>
<td>Use to establish a single sign-on connection. This operation allows for a partner application to login on behalf of the user in to NetSuite, without the user's credentials ever going through partner servers.</td>
</tr>
<tr>
<td>update / updateList</td>
<td>Use to update one or more existing records in the system by providing new values for the fields to be updated for each record. The records to be updated are identified by either the internal or external ID and the record type.</td>
</tr>
<tr>
<td>updateInviteeStatus /</td>
<td>Allows event invitees to accept or decline NetSuite events. After invitees have responded to the event, the Event record is updated with their response.</td>
</tr>
<tr>
<td>updateInviteeStatusList</td>
<td></td>
</tr>
<tr>
<td>upsert / upsertList</td>
<td>Use to add new records and update existing records in a single operation. Records are identified by external ID and record type. If a record of the specified type with a matching external ID exists in the system, it is updated. If it does not exist, a new record is created.</td>
</tr>
</tbody>
</table>

**SOAP Web Services List Operations**

Within a single SOAP request, only one operation can be performed. For example, one add, one addList or one delete. However, a single list operation (addList, updateList, upsertList, deleteList, getList, and initializeList) lets you work with multiple record types. For example, with a single addList operation you can add 3 customers, 4 opportunities, and 1 contact.

Note that list operations process records in the order in which the records are submitted. For example, an addList operation will add all records in the order they appear in the list.

**SOAP Web Services Asynchronous Operations**

The following asynchronous equivalents are available for these list operations:

- addList: asyncAddList
- updateList: asyncUpdateList
- upsertList: asyncUpsertList
- deleteList: asyncDeleteList
- getList: asyncGetList
- search: asyncSearch
- initializeList: asyncInitializeList

You can use the checkAsyncStatus and getAsyncResult operations to track asynchronous requests. For information on asynchronous request processing, see Synchronous Versus Asynchronous Request Processing.

### add

The add operation is used to add a new instance of a record in NetSuite. It is similar to the addList operation except that it allows only one record to be added at a time. Note that to prevent duplicate records, NetSuite recommends using the alternate upsert and upsertList operations along with external IDs to add records to NetSuite.

**Note:** Although records of a particular type may be used in multiple integration scenarios, each record instance can only have a single external ID value. To maintain data integrity, only a single integrated application can set and update external ID values for each record type. External ID values for all records of each type must all come from the same external application.

### Request

The AddRequest type is used for the request. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>Record</td>
<td>The record type is an abstract type so an instance of a type that extends record must be used — such as Customer or Event.</td>
</tr>
</tbody>
</table>

### Response

The AddResponse type is used for the response. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>WriteResponse</td>
<td>Contains details on the status of the operation and a reference to the created record.</td>
</tr>
</tbody>
</table>

### Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault
Sample Code

**SOAP Request**
In this example, a single customer record is added.

![SOAP Request example](image)

**SOAP Response**
In the response, notice that the internal ID for the record added is returned.

![SOAP Response example](image)

**C#**

```csharp
private void addCustomer()
{
    // This operation requires a valid session
    this.login( true );

    Customer customer = new Customer();

    // Set entityId, company name, and email
    if (!"true".Equals( _dataCollection["promptForFieldValues"] ) )
    {
        _out.writeLn( "Please enter the following customer information. " +
           "Note that some fields have already been populated. " );
        _out.write( "Entity name: " );
        customer.entityId = _out.readLn();
        _out.write( "Company name: " );
        customer.companyName = _out.readLn();
    }
```
customer.email = _out.readLn();
_customer.email = _out.readLn();
RecordRef salesRep = new RecordRef();
salesRep.internalId = _out.readLn();
//salesRep.type = RecordType.contact;
//salesRep.typeSpecified = true;
customer.salesRep = salesRep;
}
else
{
    customer.entityId = "XYZ Inc";
customer.companyName = "XYZ, Inc.";
customer.email = "bsanders@yahoo.com";
}

// Set email preference
_customer.emailPreference = CustomerEmailPreference._HTML;
customer.emailPreferenceSpecified = true;

// Set entity status. The nsKey can be obtained from Setup > SFA > Customer Statuses.
// The default status is "Closed Won" which has an nsKey of 13
RecordRef status = new RecordRef();
if ("true".Equals(_dataCollection["promptForFieldValues"]))
{
    _out.write("Entity status nsKey (press enter for default value of Closed Won): ");
    String statusKey = _out.readLn();
    if (statusKey.Equals(""))
    {
        status.internalId = "13";
    }
    else
    {
        status.internalId = statusKey;
    }
}
else
{
    status.internalId = "13";
}
customer.entityStatus = status;

// Populate the address list for this customer. You can put in as many addresses as you like.
CustomerAddressbook address = new CustomerAddressbook();
address.defaultShipping = true;
address.defaultShippingSpecified = true;
address.defaultBilling = false;
address.defaultBillingSpecified = true;
address.label = "Shipping Address";
address.addresssee = "William Sanders";
address.attention = "William Sanders";
address.addr1 = "4765 Sunset Blvd";
address.city = "San Francisco";
add

```java
address.state = "CA";
address.zip = "94131";
address.country = Country._unitedStates;

// Attach the CustomerAddressbooklist to the customer
CustomerAddressbookList addressList = new CustomerAddressbookList();
CustomerAddressbook[] addresses = new CustomerAddressbook[1];
addresses[0] = address;
adressList.addressbook = addresses;
customer.addressbookList = addressList;

// Invoke add() operation
WriteResponse response = _service.add( customer );

// Print the document id from the SOAP header
//_out.info(  //"\nThe add() operation with document id " + _service.documentInfo.nsID + " was processed " );

// Process the response
if ( response.status.isSuccess )
{
    _out.info( "\nThe following customer was added successfully:"
            + "\nkey=" + ((RecordRef) response.baseRef).internalId +
            "\nentityId=" + customer.entityId +
            "\ncompanyName=" + customer.companyName +
            "\nemail=" + customer.email +
            "\nstatusKey=" + customer.entityStatus.internalId +
            "\naddressbookList[0].label=" + customer.addressbookList.addressbook[0].label );

    // Create a second customer that’s a sub-customer of the first customer
    //
    Customer subCustomer = new Customer();

    subCustomer.entityId = "XYZ Japan";
    subCustomer.parent = recordRef;

    // Set the parent customer
    recordRef = new RecordRef();
    recordRef.internalId = ((RecordRef) response.baseRef).internalId;
    //recordRef.internalId = "125";
    subCustomer.parent = recordRef;

    // Invoke add() operation
    response = _service.add( subCustomer );

    // Process the response
    //"\nThe add() operation with job id " + _service.sessionInfo.jobID + " was processed " );
    if ( !response.status.isSuccess )
    {
        _out.info( "\nThe customer was not added:"
            + getDetails( response.status ) );
        }else
    {
        _out.info( "\nThe customer was added successfully:"
            + "\nkey=" + ((RecordRef) response.baseRef).internalId +
            "\nentityId=" + customer.entityId +
            "\ncompanyName=" + customer.companyName +
            "\nemail=" + customer.email +
            "\nstatusKey=" + customer.entityStatus.internalId +
            "\naddressbookList[0].label=" + customer.addressbookList.addressbook[0].label );
```

The following customer was added successfully:
key=" + ((RecordRef) response.baseRef).internalId + 
"\nentityId=" + customer.entityId + 
"\ncompanyName=" + customer.companyName); 
} */
}

Java

public void addCustomer() throws RemoteException, ExceededRecordCountFault,
ExceededUsageLimitFault, InsufficientPermissionFault, InvalidSessionFault {

// This operation requires a valid session
this.login(true);

Customer customer = new Customer();

// Set entityId, company name, and email
if ("true".equals(_properties.getProperty("promptForFieldValues"))) {
    _console
        .writeLn("nPlease enter the following customer information. 
        + "Note that some fields have already been populated. ");
    _console.write("Entity name: ");
    customer.setEntityId(_console.readLn());
    _console.write("Company name: ");
    customer.setCompanyName(_console.readLn());
    _console.write("E-mail: ");
    customer.setEmail(_console.readLn());
} else {
    customer.setEntityId("XYZ Inc");
    customer.setCompanyName("XYZ, Inc.");
    customer.setEmail("bsanders@yahoo.com");
}

// Set email preference.
customer.setEmailPreference(CustomerEmailPreference._hTML);

// Set entity status. The nsKey can be obtained from Setup > SFA >
// Customer Statuses.
// The default status is "Closed Won" which has an nsKey of 13
RecordRef status = new RecordRef();
if ("true".equals(_properties.getProperty("promptForFieldValues"))) {
    _console.write("Entity status nsKey (press enter for default value of Closed Won): ");
    String statusKey = _console.readLn();
    if (statusKey.equals("")) {
        status.setInternalId("13");
    } else {
        status.setInternalId(statusKey);
    }    
}
} } else {
    status.setInternalId("13");
}

customer.setEntityStatus(status);

// Populate the address list for this customer. You can put in as many
// addresses as you like.
CustomerAddressbook address = new CustomerAddressbook();
address.setDefaultShipping(Boolean.TRUE);
address.setDefaultBilling(Boolean.FALSE);
address.setLabel("Shipping Address");
address.setAddresssee("William Sanders");
address.setAttention("William Sanders");
address.setAddrl("4765 Sunset Blvd");
address.setCity("San Francisco");
address.setState("CA");
address.setZip("94131");
address.setCountry(Country._unitedStates);

// Attach the CustomerAddressbookList to the customer
CustomerAddressbookList addressList = new CustomerAddressbookList();
CustomerAddressbook[] addresses = new CustomerAddressbook[1];
adresses[0] = address;
addressList.setAddressbook(addresses);
customer.setAddressbookList(addressList);

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

// Print the document id from the SOAP header
// _console.info(
// "\nThe add() operation with document id " + _port.documentInfo.nsID +
// " was processed ");

// Process the response
if (response.getStatus().isIsSuccess()) {
    _console.info("\nThe following customer was added successfully:
    + "\nkey="
    + ((RecordRef) response.getBaseRef()).getInternalId()
    + "\nentityId="
    + customer.getEntityId()
    + "\ncompanyName="
    + customer.getCompanyName()
    + "\nemail="
    + customer.getEmail()
    + "\nstatusKey="
    + customer.getEntityStatus().getInternalId()
    + "\naddressbookList[0].label="
    + customer.getAddressbookList().getAddressbook(0)
    .getLabel());
} else {
    _console.error("The customer was not added:", true);
    _console.error(getStatusDetails(response.getStatus()));
}
addList

The addList operation is used to add one or more new instances of a record to NetSuite. If there are multiple records, they can either be of the same record type or different record types. For example, it's possible to add a customer and a contact within a single request using this operation. However, each record entered must have a unique signature. Adding two records with the same signature results in a SOAP fault. The signature consists of parameters required to identify a record as unique.

For example, in the case of entities, a record is uniquely identified by its name, its type and its parent hierarchy. So you could have two records with the same entityId (or name) belonging to two different record types as follows:

Customer (the type):
John Smith
MyCompany: John Smith

Contact
John Smith

But a second record such as the following would be invalid:

Contact
John Smith

Note: An asynchronous equivalent is available for this operation, asyncAddList. For information about asynchronous request processing, see Synchronous Versus Asynchronous Request Processing.

Note that to prevent duplicate records, NetSuite recommends using the alternate upsert and upsertList operations when adding records to NetSuite.

Request

The AddListRequest type is used for the request.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>record[]</td>
<td>Record</td>
<td>Contains an array of record objects. The record type is an abstract type so an instance of a type that extends record must be used—such as Customer or Event.</td>
</tr>
</tbody>
</table>

Response

The AddListResponse type is used for the response.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>response[]</td>
<td>WriteResponse</td>
<td>Contains an array of WriteResponse objects, each of which contains details on the status of that add operation and a reference to that created record.</td>
</tr>
</tbody>
</table>
Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In the following example, two customer records are added. When using the addList operation, you can submit two different record types in the same request.

```xml
<soap:Body>
<platformMsgs:addList>
  <platformMsgs:record xsi:type="listRel:Customer">
    <listRel:entityId>Shutter Fly</listRel:entityId>
    <listRel:companyName>Shutter Fly, Inc</listRel:companyName>
    <listRel:unsubscribe>false</listRel:unsubscribe>
  </platformMsgs:record>
  <platformMsgs:record xsi:type="listRel:Customer">
    <listRel:entityId>GNC</listRel:entityId>
    <listRel:companyName>GNC Corp</listRel:companyName>
    <listRel:unsubscribe>false</listRel:unsubscribe>
  </platformMsgs:record>
</platformMsgs:addList>
</soap:Body>
```

SOAP Response

In the response, notice that the internal ID for each record added is returned.

```xml
<soapenv:Body>
<addListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<writeResponseList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <writeResponse>
    <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com">
      <baseRef internalId="980" type="customer" xsi:type="ns2:RecordRef">
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
    </writeResponse>
  </writeResponseList>
</addListResponse>
</soapenv:Body>
```
Java Sample

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

    Customer cust1 = new Customer();
    cust1.setEntityId("Shutter Fly");
    cust1.setCompanyName("Shutter Fly, Inc");
    cust1.setUnsubscribe(false);

    Customer cust2 = new Customer();
    cust2.setEntityId("GNC");
    cust2.setCompanyName("GNC Corp");
    cust2.setUnsubscribe(false);

    Customer[] customers = new Customer[2];
    customers[0] = cust1;
    customers[1] = cust2;

    WriteResponseList responseList = _port.addList(customers);
}
```

attach / detach

The attach and detach operations can be used to define or remove a relationship between two records. For example, a Contact record can be associated with a Partner record, or an Opportunity record can be associated with a Customer record.

**Important**: A user error is thrown if you attempt to attach files or records that do not exist.

You can also use the attach / detach operations to attach or detach a file to or from a record. Any file that is in the NetSuite file cabinet, for example an MS Word or Excel file or a PDF can be attached to any record other than a custom record.

Note that when attaching Contacts to other entity records, the Contact's role can also be specified during the request. Contact Roles are roles available in a user defined list at List > Relationships > Contacts. This list has been exposed as ContactRole in accounting.xsd.

Note that to prevent duplicate records, NetSuite recommends using the alternate upsert and upsertList operations along with external ids to add records to NetSuite.
**Note:** Contact records can be attached to all entity records expect for other Contact or Group records.

The following table lists all records that support the attach/detach operations. It also lists which records can accept file attachments as well as which records can accept Contact records as attachments.

<table>
<thead>
<tr>
<th>Record Type</th>
<th>Accepts File Attachments</th>
<th>Accepts Contact Record Attachments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Check</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Custom Transaction</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Expense Report</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Inventory Adjustment</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Item Fulfillment</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Journal Entry</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Intercompany Transfer Order</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Opportunity</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Transfer Order</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Sales Order</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Customer Payment</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Return Authorization</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Credit Memo</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Cash Refund</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Estimate</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Invoice</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Cash Sale</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Purchase Order</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Requisition</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Customer Payment</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Customer Refund</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Customer Deposit</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Vendor Bill</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Vendor Credit</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Vendor Return Authorization</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Work Order</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Record Type</td>
<td>Accepts File Attachments</td>
<td>Accepts Contact Record Attachments</td>
</tr>
<tr>
<td>---------------------</td>
<td>--------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Work Order Issue</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Work Order Close</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Work Order Completion</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Entities**

<table>
<thead>
<tr>
<th>Entity</th>
<th>Accepts File Attachments</th>
<th>Accepts Contact Record Attachments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Contact</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Employee</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Partner</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Vendor</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Project</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Group</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Although you cannot currently attach Contact records to Group, you can attach members to create a static group.

**Activities**

<table>
<thead>
<tr>
<th>Activity</th>
<th>Accepts File Attachments</th>
<th>Accepts Contact Record Attachments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Event</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Phone Call</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Project Task</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Support**

<table>
<thead>
<tr>
<th>Support Type</th>
<th>Accepts File Attachments</th>
<th>Accepts Contact Record Attachments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Case</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Issue</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Marketing**

<table>
<thead>
<tr>
<th>Marketing Type</th>
<th>Accepts File Attachments</th>
<th>Accepts Contact Record Attachments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Campaign</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

**Request (attach)**

The AttachRequest type is used for this request. It contains the following fields:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>attachReference</td>
<td>AttachReference</td>
<td></td>
</tr>
</tbody>
</table>

**Request (detach)**

The DetachRequest type is used for this request. It contains the following fields:
### Response (attach)

The AttachResponse type is used for this response. It contains the following fields:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>WriteResponse</td>
<td></td>
</tr>
</tbody>
</table>

### Response (detach)

The DetachResponse type is used for this response. It contains the following fields:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>WriteResponse</td>
<td></td>
</tr>
</tbody>
</table>

### Faults

This operation can throw one of the following faults. See [SOAP Fault Status Codes](https://www.oracle.com) for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

### Sample Code

**SOAP Request (attach)**

```xml
<attach xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <attachReference xsi:type="ns1:AttachContactReference" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com">
    <ns1:attachTo internalId="176" type="customer" xsi:type="ns1:RecordRef">
      <ns1:name xsi:type="xsd:string">Adelina Shonkwiler</ns1:name>
    </ns1:attachTo>
    <ns1:contact internalId="1467" xsi:type="ns1:RecordRef"/>
    <ns1:contactRole internalId="-10" xsi:type="ns1:RecordRef">
      <ns1:name xsi:type="xsd:string">Primary Contact</ns1:name>
    </ns1:contactRole>
  </attachReference>
</attach>
```
SOAP Request (detach)

```xml
<detach xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <attachReference xsi:type="ns1:AttachBasicReference"
      xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com">
    <ns1:attachTo internalId="176" type="customer" xsi:type="ns1:RecordRef">
      <ns1:name xsi:type="xsd:string">Adelina Shonkwiler</ns1:name>
    </ns1:attachTo>
    <ns1:attachedRecord internalId="1467" type="contact" xsi:type="ns1:RecordRef"/>
  </attachReference>
</detach>
```

SOAP Response (attach)

```xml
<attachResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <writeResponse>
    <ns1:status isSuccess="true"
        xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com">
      <baseRef internalId="176" type="customer" xsi:type="ns2:RecordRef"
          xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">
        <ns2:name>Adelina Shonkwiler</ns2:name>
      </baseRef>
    </ns1:status>
  </writeResponse>
</attachResponse>
```

SOAP Response (detach)

```xml
<detachResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <writeResponse>
    <ns1:status isSuccess="true"
        xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com">
      <baseRef internalId="176" type="customer" xsi:type="ns2:RecordRef"
          xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">
        <ns2:name>Adelina Shonkwiler</ns2:name>
      </baseRef>
    </ns1:status>
  </writeResponse>
</detachResponse>
```

Java (attach operation)

```java
public void attach() throws Exception {
    RecordRef contactRef = new RecordRef();
    contactRef.setInternalId("1467");
    contactRef.setType(RecordType.contact);

    RecordRef contactRoleRef = new RecordRef();
    contactRoleRef.setInternalId("-10");
}
```
contactRoleRef.setName("Primary Contact");

RecordRef customerRef = new RecordRef();
customerRef.setInternalId("176");
customerRef.setType(RecordType.customer);

AttachContactReference attachRef = new AttachContactReference();
attachRef.setContact(contactRef);
attachRef.setAttachTo(customerRef);
attachRef.setContactRole(contactRoleRef);

WriteResponse attachResponse = sessMgr.getPort().attach(attachRef);
}

Java (detach operation)

public void detach() throws Exception
{
    RecordRef contactRef = new RecordRef();
    contactRef.setInternalId("1467");
    contactRef.setType(RecordType.contact);

    RecordRef customerRef = new RecordRef();
    customerRef.setInternalId("176");
    customerRef.setType(RecordType.customer);

    AttachContactReference detachRef = new AttachContactReference();
detachRef.setContact(contactRef);
detachRef.setAttachTo(customerRef);

    WriteResponse detachResponse = sessMgr.getPort().detach(detachRef);
}

changePasswordOrEmail

The changePasswordOrEmail operation is supported in the 2012.1 and earlier endpoints. Use this operation to change \textbf{either} a user’s email address or password. You cannot use this operation to change both the email address and password in a single call.

\textbf{Warning:} If you upgrade to the 2012.2 endpoint or a later version, any existing calls to changeEmailOrPassword will no longer work. Instead, you must use the \textit{changeEmail} operation to change a user’s email address, and you must use \textit{changePassword} to change a user’s password.

\textbf{Important:} When using the changePasswordOrEmail operation to change email addresses, be aware that the addresses are not changed immediately. The old email address remains in effect until the new email address is validated by the user, through a validation link sent to the new address. A notification is also sent to the old address. This behavior is the same as when users change their email addresses through the user interface.

For details on how to change a user’s email address or password using the NetSuite user interface, see the following topics:
Your User Credentials
Change Email Link
Change Password Link

Request

The ChangePasswordOrEmailRequest type is used for the request.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>changePasswordOrEmailCredentials</td>
<td>ChangePasswordOrEmailCredentials</td>
<td></td>
</tr>
</tbody>
</table>

The ChangePasswordOrEmailCredentials type takes the following fields:

- currentPassword
- newEmail
- newEmail2
- newPassword
- newPassword2
- justThisAccount

Response

The ChangePasswordOrEmailResponse type is used for the response. It does not contain any fields.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InsufficientPermissionFault
- InvalidAccountFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- UnexpectedErrorFault
- InvalidVersionFault

Example — Changing an Email Address

The following example shows how to use the changeEmailOrPassword operation to change a user’s email address.

Java

```java
public void test_ChangePwdOrEmail() throws Exception {
```

SuiteTalk SOAP Web Services Platform Guide
Example — Changing a Password

The following example shows how to change a password. Note that with this approach, you have to declare values for NewEmail and NewEmail2 — but you should specify empty strings. These fields are ignored, but you must declare them, or the operation fails.

Java

```java
public void changePassword() throws Exception {
    ChangePasswordOrEmailCredentials newCredentials = new ChangePasswordOrEmailCredentials();
    newCredentials.setCurrentPassword("currentPwd");
    newCredentials.setNewPassword("newPwd");
    newCredentials.setNewPassword2("newPwd");
    newCredentials.setNewEmail("" ); // NewEmail is mandatory, and you must specify an empty string for the operation to complete successfully
    newCredentials.setNewEmail2("" ); // NewEmail2 is mandatory, and you must specify an empty string for the operation to complete successfully
    sessMgr.loginWithEmail("oldEmail@ws.com");
    ChangePasswordOrEmailCredentials c = new ChangePasswordOrEmailCredentials();
    c.setCurrentPassword("xxxxxx");
    c.setNewEmail("newEmail@ws.com");
    c.setNewEmail2("newEmail@ws.com");
    sessMgr.getPort().changePasswordOrEmail(c);
    sessMgr.loginWithEmail("newEmail@ws.com");
}
```
changePasswordOrEmail

```java
SessionResponse response = sessMgr.getPort().changePasswordOrEmail(newCredentials);
```

### SOAP Request

```xml
<changePasswordOrEmail xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <changePasswordOrEmailCredentials>
    <ns7:currentPassword xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">currentPwd</ns7:currentPassword>
    <ns8:newEmail xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com"></ns8:newEmail>
    <ns9:newEmail2 xmlns:ns9="urn:core_2017_1.platform.webservices.netsuite.com"></ns9:newEmail2>
    <ns10:newPassword xmlns:ns10="urn:core_2017_1.platform.webservices.netsuite.com">newPwd</ns10:newPassword>
    <ns11:newPassword2 xmlns:ns11="urn:core_2017_1.platform.webservices.netsuite.com">newPwd</ns11:newPassword2>
  </changePasswordOrEmailCredentials>
</changePasswordOrEmail>
```

### changeEmail

Use the changeEmail operation to change a user's email address.

This topic provides details about how to change a user's email address with SOAP web services code. For details about how to make this change in the NetSuite user interface, see the help topics Change Email Link and Your User Credentials.

**Important:** This operation is supported as of the 2012.2 endpoint. The 2012.2 and later endpoints support changeEmail and changePassword operations that replace changePasswordOrEmail. Endpoints 2012.2 and later do not support the changePasswordOrEmail operation.

**Important:** Be aware that a user's email address is not changed immediately. The old email address remains in effect until the new email address is validated by the user, through a validation link sent to the new address. A notification is also sent to the old address. This behavior is the same as when users change their email addresses through the user interface.

### Request

The ChangeEmailRequest type is used for the request.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>changeEmail</td>
<td>ChangeEmail</td>
<td></td>
</tr>
</tbody>
</table>

The ChangeEmail type takes the following fields:

- currentPassword
- newEmail
- newEmail2
Response

The ChangeEmailResponse type is used for the response. It does not contain any fields.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InsufficientPermissionFault
- InvalidAccountFault
- InvalidCredentialsFault
- InvalidVersionFault
- ExceededRequestLimitFault
- UnexpectedErrorFault
- UserError

Sample Code

SOAP Request

```xml
<changeEmail xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <changeEmail>
        <ns6:currentPassword xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.com">xxxxxx</ns6:currentPassword>
        <ns7:newEmail xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">newEmail@tester.com</ns7:newEmail>
        <ns8:newEmail2 xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com">newEmail@tester.com</ns8:newEmail2>
        <ns9:justThisAccount xmlns:ns9="urn:core_2017_1.platform.webservices.netsuite.com">true</ns9:justThisAccount>
    </changeEmail>
</changeEmail>
```

SOAP Response

```xml
<changeEmailResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <sessionResponse>
        <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com/>
    </sessionResponse>
</changeEmailResponse>
```
Java

```java
public void test_ChangeEmail() throws Exception {
    sessMgr.loginWithEmail("oldEmail@ws.com");
    ChangeEmail c = new ChangeEmail();
    c.setCurrentPassword("xxxxxx");
    c.setNewEmail("newEmail@ws.com");
    c.setNewEmail2("newEmail@ws.com");
    c.setJustThisAccount(true);
    sessMgr.getPort().changeEmail(c);
    sessMgr.loginWithEmail("newEmail@ws.com");
}
```

changePassword

Use the changePassword operation to change a user's NetSuite password.

This topic provides details about how to change a user's password with SOAP web services code. For details about how to make this change in the NetSuite user interface, see the help topics Change Password Link and Your User Credentials.

Request

The ChangePasswordRequest type is used for the request.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>changePassword</td>
<td>ChangePassword</td>
<td></td>
</tr>
</tbody>
</table>

The ChangePassword type takes the following fields:

- currentPassword
- newPassword
- newPassword2

Response

The ChangePasswordResponse type is used for the response. It does not contain any fields.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InsufficientPermissionFault
- InvalidAccountFault
- InvalidCredentialsFault
Sample Code

SOAP Request

```xml
<soapenv:Body>
  <changePassword xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <ns6:currentPassword xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.com">password1</ns6:currentPassword>
    <ns7:newPassword xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">password2</ns7:newPassword>
    <ns8:newPassword2 xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com">password2</ns8:newPassword2>
  </changePassword>
</soapenv:Body>
```

SOAP Response

```xml
<soapenv:Body>
  <changePasswordResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <sessionResponse>
      <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
    </sessionResponse>
  </changePasswordResponse>
</soapenv:Body>
```

Java

```java
ChangePassword changePW = new ChangePassword();
changePW.setCurrentPassword("password1");
changePW.setNewPassword("password2");
changePW.setNewPassword2("password2");
c.getPort().changePassword(changePW);
```

checkAsyncStatus

See checkAsyncStatus in Synchronous Versus Asynchronous Request Processing for details.
The delete operation is used to delete an instance of a record. It is similar to the deleteList operation, except that it permits only one record to be deleted per request.

### Request

The DeleteRequest type is used for the request. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>recordRef</td>
<td>RecordRef</td>
<td>Identifies the record to be deleted.</td>
</tr>
<tr>
<td>deletionReason</td>
<td>DeletionReason</td>
<td>Identifies a deletion reason and deletion reason memo. This parameter supports the Use Deletion Reasons feature, which requires users to provide reasons when they delete transactions. However, even when this feature is not enabled, you must use this parameter with every delete request. (You can provide a value of null in such cases.) For more details, see Deletion Reason Usage Notes.</td>
</tr>
</tbody>
</table>

### Deletion Reason Usage Notes

Note the following about the deletionReason parameter:

- The deletionReason complex type includes two fields: deletionReasonCode and deletionReasonMemo. The deletionReasonCode must identify a deletion reason that is listed at Setup > Accounting > Accounting Lists. If the Use Deletion Reasons feature is enabled and you use the deletionReasonCode to identify a code that does not exist, the request fails with an INVALID_REF_KEY error.
- Deletion reasons can be saved only for transactions. However, in SOAP web services, you must use the deletionReason parameter even when referencing other record types, and even when the Use Deletion Reasons feature is not enabled. For situations where it is not appropriate to identify a deletion reason, pass in a value of null.
- Even when when a deletion reason is required, you can use a value of null. In these cases, the system automatically populates the deletion reason fields with default values. These defaults are: Other for deletionReasonCode and This transaction was deleted by script or web service for deletionReasonMemo.

The deletionReason complex type is defined in the core XSD.

For more details about the Use Deletion Reasons feature, see the help topic Recording a Reason for Deleting a Transaction.

### Response

The DeleteResponse type is used for the response. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>WriteResponse</td>
<td>Contains details on the status of the delete operation and a reference to the deleted record.</td>
</tr>
</tbody>
</table>
Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

**Note:** If you attempt to delete a record that does not exist, the system returns an INVALID_KEY_OR_REF error.

Sample Code

The following example shows how to delete a cash sale transaction.

**C#**

```csharp
private void deleteCashSale()
{
    RecordRef myCashSale = new RecordRef();
    myCashSale.internalId = "1745";
    myCashSale.type = RecordType.cashSale;
    myCashSale.typeSpecified = true;

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

    DeletionReason myDeletionReason = new DeletionReason();
    myDeletionReason.deletionReasonCode = myDeletionReasonCode;
    myDeletionReason.deletionReasonMemo = "Per John in Accounting";

    _service.delete(myCashSale, myDeletionReason);
}
```

**SOAP Request**

```xml
<soap:Body>
    <delete xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
        <baseRef type="cashSale" internalId="1745" xsi:type="q1:RecordRef"
            xmlns:q1="urn:core_2017_1.platform.webservices.netsuite.com"/>
        <deletionReason>
```

SuiteTalk SOAP Web Services Platform Guide
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.webservices.netsuite.com"/>
      <baseRef xsi:type="platformCore:RecordRef" type="cashSale" internalId="1745" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
    </writeResponse>
  </deleteResponse>
</soapenv:Body>
```

deleteList

The deleteList operation is used to delete one or more record instances. The record instances can be of various record types. For example, it is possible to delete a customer and a contact within a single request by using this operation.

**Note:** An asynchronous equivalent is available for this operation, `asyncDeleteList`. For information about asynchronous request processing, see Synchronous Versus Asynchronous Request Processing.

Request

The DeleteListRequest type is used for the request. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>record[]</td>
<td>RecordRef</td>
<td>An array of recordRef objects that specify the records to be deleted.</td>
</tr>
<tr>
<td>deletionReason</td>
<td>DeletionReason</td>
<td>Identifies a deletion reason and deletion reason memo. This parameter supports the Use Deletion Reasons feature, which requires users to provide reasons when they delete transactions. However, even when this feature is not enabled, you must use this parameter with every deleteList request. (You can provide a value of null in such cases.) For more details, see Deletion Reason Usage Notes.</td>
</tr>
</tbody>
</table>

Deletion Reason Usage Notes

Note the following about the deletionReason parameter:
A single deletion reason applies to the entire array of records being deleted. If some of the records are transactions and some are not, then the deletion reason still applies to the transactions. It is ignored for the other records.

The deletionReason complex type includes two fields: deletionReasonCode and deletionReasonMemo. The deletionReasonCode must identify a deletion reason that is listed at Setup > Accounting > Accounting Lists. If the Use Deletion Reasons feature is enabled and you use the deletionReasonCode to identify a code that does not exist, the request fails with an INVALID_REF_KEY error.

Deletion reasons can be saved only for transactions. However, in SOAP web services, you must use the deletionReason parameter even when referencing other record types, and even when the Use Deletion Reasons feature is not enabled. For situations where it is not appropriate to identify a deletion reason, pass in a value of null.

Even when a deletion reason is required, you can use a value of null. In these cases, the system automatically populates the deletion reason fields with default values. These defaults are: Other for deletionReasonCode and This transaction was deleted by script or web service for deletionReasonMemo.

The deletionReason complex type is defined in the core XSD.

For more details about the Use Deletion Reasons feature, see the help topic Recording a Reason for Deleting a Transaction.

Response

The DeleteListResponse type is used for the response. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>response[]</td>
<td>WriteResponse</td>
<td>Contains an array of WriteResponse objects, each of which contains details on the status of the delete operation and a reference to the deleted record.</td>
</tr>
</tbody>
</table>

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

The following examples show various ways of using the deleteList operation.
C#

The following example shows how to delete two cash sale transactions. Both transactions have the same deletion reason.

```csharp
private void deleteCashSales()
{
    RecordRef myCashSale = new RecordRef();
    myCashSale.internalId = "1727";
    myCashSale.type = RecordType.cashSale;
    myCashSale.typeSpecified = true;

    RecordRef myCashSale2 = new RecordRef();
    myCashSale2.internalId = "1728";
    myCashSale2.type = RecordType.cashSale;
    myCashSale2.typeSpecified = true;

    RecordRef myDeletionReasonCode = new RecordRef();
    myDeletionReasonCode.internalId = "2";

    DeletionReason myDeletionReason = new DeletionReason();
    myDeletionReason.deletionReasonCode = myDeletionReasonCode;
    myDeletionReason.deletionReasonMemo = "Per Scott";

    RecordRef[] myCashSaleRefs = new RecordRef[2];
    myCashSaleRefs[0] = myCashSale;
    myCashSaleRefs[1] = myCashSale2;

    _service.deleteList(myCashSaleRefs, myDeletionReason);
}
```

The following example shows how to delete two records of different types: an employee record and a calendar event record. Because it's not possible to set a deletion reason for either of those two types of records, the deletion reason used is null.

```csharp
private void deleteVariousRecords()
{
    RecordRef myEmployee = new RecordRef();
    myEmployee.internalId = "360";
    myEmployee.type = RecordType.employee;
    myEmployee.typeSpecified = true;

    RecordRef myCalendarEvent = new RecordRef();
    myCalendarEvent.internalId = "381";
    myCalendarEvent.type = RecordType.calendarEvent;
    myCalendarEvent.typeSpecified = true;

    RecordRef[] myRecords = new RecordRef[2];
    myRecords[0] = myEmployee;
    myRecords[1] = myCalendarEvent;

    _service.deleteList(myRecords, null);
}
```
SOAP Request

The following SOAP request shows how to delete an employee and calendar event record.

```xml
<soap:Body>
  <deleteList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <baseRef type="employee" internalId="360" xsi:type="q1:RecordRef"
      xmlns:q1="urn:core_2017_1.platform.webservices.netsuite.com"/>
    <baseRef type="calendarEvent" internalId="381" xsi:type="q2:RecordRef"
      xmlns:q2="urn:core_2017_1.platform.webservices.netsuite.com"/>
  </deleteList>
</soap:Body>
```

SOAP Response

In the response, the status for each item in the request is returned. If a problem exists with one of the items, creating an error, the other records are still processed.

```xml
<soapenv:Body>
  <deleteListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <writeResponseList>
      <platformCore:status isSuccess="true"
        xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      <writeResponse>
        <platformCore:status isSuccess="true"
          xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
        <baseRef xsi:type="platformCore:RecordRef" type="employee" internalId="360"
          xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
      <writeResponse>
        <platformCore:status isSuccess="true"
          xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
        <baseRef xsi:type="platformCore:RecordRef" type="calendarEvent" internalId="381"
          xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </writeResponse>
    </writeResponseList>
  </deleteListResponse>
</soapenv:Body>
```

get

The get operation is used to retrieve a record by providing the unique id that identifies that record.

Request

The getRequest type is used for the request. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>recordRef</td>
<td>RecordRef</td>
<td>A recordRef object that specifies the id of the record to be retrieved.</td>
</tr>
</tbody>
</table>
Response

The getResponse type is used for the response. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this operation. All applicable errors or warnings are listed within this type.</td>
</tr>
<tr>
<td>record</td>
<td>Record</td>
<td>A record that represents the specified id. The actual record returned needs to be a type that extends the abstract type Record.</td>
</tr>
</tbody>
</table>

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Usage Notes for Custom Fields Generated from Custom Segments

The get operation supports the usage of unified scripts IDs of custom segments. If the unified script ID is used on a custom segment, it is returned, if the legacy one is used, it is returned.

As of 2019.1, any new custom segments that you create automatically use the unified ID.

SOAP Response with Unified Script ID Used on the Custom Segment

```xml
<soapenv:Body>
  <getResponse xmlns="urn:messages_2018_2.platform.webservices.netsuite.com">
    <readResponse>
      <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2018_2.platform.webservices.netsuite.com"/>
      <record internalId="107" xsi:type="listRel:Customer" xmlns:listRel="urn:relationships_2018_2.lists.webservices.netsuite.com"/>
      <listRel:customForm internalId="3" xmlns:listRel="urn:relationships_2018_2.lists.webservices.netsuite.com">Customer_Preferred=T_Store=F</listRel:customForm>
      <platformCore:name>Customer_Preferred=T_Store=F</platformCore:name>
  </readResponse>
</getResponse>
```
SOAP Request – Return Customer Record

In this example, a single customer record is retrieved. Note that the internal ID for the specific instance of the record and the record type (customer) must be specified.

```xml
<soapenv:Body>
  <get
      xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <baseRef
        internalId="107" type="customer" xsi:type="ns7:RecordRef"
        xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com"/>
  </get>
</soapenv:Body>
```

SOAP Response

```xml
<soapenv:Body>
  <getResponse
      xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <readResponse>
      <platformCore:status
          isSuccess="true"
          xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
      <record
          internalId="107" xsi:type="listRel:Customer"
          xmlns:listRel="urn:relationships_2017_1.lists.webservices.netsuite.com">
        <listRel:customForm internalId="3"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
          <platformCore:name>Customer_Preferred=T_Store=F</platformCore:name>
        </listRel:customForm>
        <listRel:entityId>Tim Harris</listRel:entityId>
        <listRel:isPerson>false</listRel:isPerson>
        <listRel:companyName>Glenrock General Hospital</listRel:companyName>
        <listRel:entityStatus internalId="13"
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
          <platformCore:name>CUSTOMER-Closed Won</platformCore:name>
        </listRel:entityStatus>
        <listRel:phone>555-394-3830</listRel:phone>
        <listRel:email>tim@example.com</listRel:email>
        ...
    </record>
  </readResponse>
</getResponse>
```

...[more fields]...
private void getCustomer() {
    // This operation requires a valid session
    this.login( true );

    // Prompt for the nsKey
    _out.write( "nsKey for record to be retrieved: " );
    String nsKey = _out.readLn();

    // Invoke the get() operation to retrieve the record
    RecordRef recordRef = new RecordRef();
    recordRef.internalId = nsKey;
    recordRef.type = RecordType.customer;
    recordRef.typeSpecified = true;

    ReadResponse response = _service.get( recordRef );

    // Process response from get() operation
    _out.info( "Record returned from get() operation: " );
    if ( !response.status.isSuccess )
    {
        _out.info(
            "ERROR: " +
            getStatusDetails( response.status ) );
    }
    else
    {
        Customer customer = (Customer) response.record;
        _out.info(
            "nsKey=" + customer.internalId + ", " +
            "entityId=" + customer.entityId +
            (customer.companyName == null ? "" : "companyName=") +
            customer.companyName + "");
    }
Java

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

    // Prompt for the nsKey
    _console.write("nsKey for record to be retrieved: ");
    String nsKey = _console.readInt();

    // Invoke the get() operation to retrieve the record
    RecordRef recordRef = new RecordRef();
    recordRef.setInternalId(nsKey);
    recordRef.setType(RecordType.customer);
    ReadResponse response = _port.get(recordRef);

    // Process response from get() operation
    _console.info("Record returned from get() operation:");
    if (!response.getStatus().isIsSuccess()) {
        _console.info("ERROR: " + getStatusDetails(response.getStatus()));
    } else {
        Customer customer = (Customer) response.getRecord();
        _console.info("nsKey=
                       "+ customer.getInternalId()
                       + ", "
                       + "entityId="
                       + customer.getEntityId()
                       + (customer.getCompanyName() == null ? ""
                       : ("\ncompanyName=" + customer
                       .getCompanyName()))
                       + (customer.getStage() == null ? ""
                       : ("\nstage=" + customer.getStage()))
                       + (customer.getEmail() == null ? ""
                       : ("\nemail=" + customer.getEmail()))
                       + (customer.getPhone() == null ? ""
                       : ("\nphone=" + customer.getPhone()))
                       + "isInactive="
                       + customer.isInactive()
                       + (customer.getDateCreatedSpecified() ? ""
                       : ("\ndateCreated=" +
                       customer.getDateCreated().ToShortDateString()));
    }
}
```
getAll

The `getAll` operation is used to retrieve a list of all records of the specified type. Records that support the `getAll` operation are listed in the `GetAllRecordType`, as defined in the `platformCoreType` system constants XSD file.

**Note:** You cannot use the search operation to retrieve state values. You must use the `getAll` operation. The `getAll` operation will return all states, not the legal ones for your default country. Also note that the country and state must match on the address.

Request

The `getAllRequest` type is used for the request. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>recordType</td>
<td>GetAllRecordType</td>
<td>Specify the record type.</td>
</tr>
</tbody>
</table>

Response

The `getList` response type is used for the response. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this operation. All applicable errors or warnings are listed within this type.</td>
</tr>
<tr>
<td>recordList</td>
<td>Record[]</td>
<td>A list of records that correspond to the specified ids. The actual records returned need to be of a type that extends the abstract type Record.</td>
</tr>
</tbody>
</table>

Faults

This operation can throw one of the following faults. See [SOAP Fault Status Codes](#) for more information on faults.

- `InvalidSessionFault`
- `InvalidCredentialsFault`
- `ExceededRequestLimitFault`
- `ExceededUsageLimitFault`
- `ExceededRecordCountFault`
- `ExceededRequestSizeFault`
- `UnexpectedErrorFault`
Sample Code

The following sample shows how to retrieve the list of all Currency records.

Java

```java
public void getAll() throws Exception {
    c.getAll(GetAllRecordType.currency);
}
```

SOAP Request

```xml
<soapenv:Body>
    <getAll xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
        <record recordType="currency"/>
    </getAll>
</soapenv:Body>
```

SOAP Response

```xml
<soapenv:Body>
    <getAllResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
        <platformCore:getAllResult xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
            <platformCore:status isSuccess="true"/>
            <platformCore:totalRecords>2</platformCore:totalRecords>
            <platformCore:recordList>
                <platformCore:record internalId="1" xsi:type="listAcct:Currency" xmlns:listAcct="urn:accounting_2017_1.lists.webservices.netsuite.com">
                    <listAcct:name>US USD</listAcct:name>
                    <listAcct:symbol>USD</listAcct:symbol>
                    <listAcct:isBaseCurrency>true</listAcct:isBaseCurrency>
                    <listAcct:isInactive>false</listAcct:isInactive>
                    <listAcct:overrideCurrencyFormat>false</listAcct:overrideCurrencyFormat>
                    <listAcct:displaySymbol>$</listAcct:displaySymbol>
                    <listAcct:symbolPlacement>_beforeNumber</listAcct:symbolPlacement>
                    <listAcct:locale>_unitedStatesEnglish</listAcct:locale>
                    <listAcct:formatSample>$1,234.56</listAcct:formatSample>
                    <listAcct:exchangeRate>1.0</listAcct:exchangeRate>
                    <listAcct:currencyPrecision>_two</listAcct:currencyPrecision>
                </platformCore:record>
                <platformCore:record internalId="2" xsi:type="listAcct:Currency" xmlns:listAcct="urn:accounting_2017_1.lists.webservices.netsuite.com">
                    <listAcct:name>UK GBP</listAcct:name>
                    <listAcct:symbol>GBP</listAcct:symbol>
                    <listAcct:isBaseCurrency>true</listAcct:isBaseCurrency>
                    <listAcct:isInactive>false</listAcct:isInactive>
                    <listAcct:overrideCurrencyFormat>false</listAcct:overrideCurrencyFormat>
                    <listAcct:displaySymbol>£</listAcct:displaySymbol>
                    <listAcct:symbolPlacement>_beforeNumber</listAcct:symbolPlacement>
                    <listAcct:locale>_unitedKingdomEnglish</listAcct:locale>
                    <listAcct:formatSample>£1,234.56</listAcct:formatSample>
                    <listAcct:exchangeRate>1.0</listAcct:exchangeRate>
                    <listAcct:currencyPrecision>_two</listAcct:currencyPrecision>
                </platformCore:record>
            </platformCore:recordList>
        </platformCore:getAllResult>
    </getAllResponse>
</soapenv:Body>
```
getAsyncResult

See Checking for Detailed Results of an Asynchronous Job in Asynchronous Request Processing for details.

getBudgetExchangeRate

On the Budget Exchange Rates table, you can maintain exchange rates between the root-parent and child subsidiaries for use in the budgeting process. Use the getBudgetExchangeRate operation to get and filter all data related to this table.

⚠️ Important: This operation can be used only in NetSuite OneWorld accounts.

In the UI, you can see the Budget Exchange Rates table by going to List > Accounting > Budget Exchange Rates. Note that in SOAP web services, this table is read-only.

For general information on the Budget Exchange Rate table, see the help topic Budget Exchange Rates.

Request

The GetBudgetExchangeRateRequest type is used for the request.
<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>budgetExchangeRateFilter</td>
<td>BudgetExchangeRateFilter</td>
<td>You can filter the returned exchange rates for a budget using this filter.</td>
</tr>
</tbody>
</table>

**BudgetExchangeRateFilter**

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>period</td>
<td>RecordRef</td>
<td>References an existing period. This argument is required.</td>
</tr>
<tr>
<td>fromSubsidiary</td>
<td>RecordRef</td>
<td>References the receiving subsidiary. This argument is optional.</td>
</tr>
<tr>
<td>toSubsidiary</td>
<td>RecordRef</td>
<td>References the originating subsidiary. This argument is optional.</td>
</tr>
</tbody>
</table>

**Response**

The GetBudgetExchangeRateResult type is used for the response.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this operation. All applicable errors or warnings are listed within this type.</td>
</tr>
<tr>
<td>budgetExchangeRateList</td>
<td>BudgetExchangeRateList</td>
<td>Returns a list of available exchange rates in a budget.</td>
</tr>
</tbody>
</table>

**BudgetExchangeRateList**

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>budgetExchangeRate</td>
<td>BudgetExchangeRate</td>
<td>References the exchange rate for a budget.</td>
</tr>
</tbody>
</table>

**BudgetExchangeRate**

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>period</td>
<td>RecordRef</td>
<td>References an existing period.</td>
</tr>
<tr>
<td>fromSubsidiary</td>
<td>RecordRef</td>
<td>References the receiving subsidiary.</td>
</tr>
<tr>
<td>toSubsidiary</td>
<td>RecordRef</td>
<td>References the originating subsidiary.</td>
</tr>
<tr>
<td>currentRate</td>
<td>double</td>
<td>References the current rate.</td>
</tr>
<tr>
<td>averageRate</td>
<td>double</td>
<td>References the average rate.</td>
</tr>
<tr>
<td>historicalRate</td>
<td>double</td>
<td>References the historical rate.</td>
</tr>
</tbody>
</table>

**Faults**

This operation can throw one of the following faults. See [SOAP Fault Status Codes](https://example.com) for more information on faults.
- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

**SOAP Request**

```xml
<?xml version='1.0' encoding='UTF-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getBudgetExchangeRate xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <budgetExchangeRateFilter>
        <ns1:period internalId="3" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"/>
        <ns2:fromSubsidiary internalId="4" xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
      </budgetExchangeRateFilter>
    </getBudgetExchangeRate>
  </soapenv:Body>
</soapenv:Envelope>
```

**SOAP Response**

```xml
<?xml version='1.0' encoding='utf-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getBudgetExchangeRateResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <platformCore:getBudgetExchangeRateResult xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
        <platformCore:status isSuccess="true"/>
        <platformCore:budgetExchangeRateList>
          <platformCore:budgetExchangeRate>
            <platformCore:period internalId="3"/>
            <platformCore:fromSubsidiary internalId="4"/>
            <platformCore:toSubsidiary internalId="1"/>
            <platformCore:currentRate>1.9586</platformCore:currentRate>
            <platformCore:averageRate>1.9586</platformCore:averageRate>
            <platformCore:historicalRate>1.9586</platformCore:historicalRate>
          </platformCore:budgetExchangeRate>
        </platformCore:budgetExchangeRateList>
      </platformCore:getBudgetExchangeRateResult>
    </getBudgetExchangeRateResponse>
  </soapenv:Body>
</soapenv:Envelope>
```
getBudgetExchangeRate

Java

```java
/* Make Record Ref out of an internalId */
public static RecordRef mrr(String internalId)
{
    RecordRef toRet = new RecordRef();
    toRet.setInternalId(internalId);
    return toRet;
}

c.getPort().getBudgetExchangeRate(new BudgetExchangeRateFilter(mrr("3"), mrr("4"), null));
```

getConsolidatedExchangeRate

On the Consolidated Exchange Rates table, you can maintain exchange rates between the root-parent and child subsidiaries. Use the getConsolidatedExchangeRate operation to get and filter all data related to this table.

**Important:** The exposure of the consolidated exchange rate record prompted the removal of the getConsolidatedExchangeRate operation from the 2017.1 endpoint. You can still use the getConsolidatedExchangeRate operation on earlier endpoints. For more information about working with consolidated exchange rates on endpoints starting from 2017.1, see the help topic Consolidated Exchange Rate.

**Important:** This operation can be used only in NetSuite OneWorld accounts.

In the UI, you can see the Consolidated Exchange Rates table by going to List > Accounting > Consolidated Exchange Rates. Note that in SOAP web services, this table is read-only.

For general information on currencies and working with the Consolidated Exchange Rate table, see the following topics:
getConsolidatedExchangeRate

Creating Currency Records

Currency Exchange Rates

**Note:** If you choose, you can set exchange rates in the Consolidated Exchange Rates table directly. See the steps under “To set exchange rates in the Consolidated Exchange Rates table” for details.

Currency Exchange Rate Integration

Currency Management

getConsolidatedExchangeRate for the Multi-Book Accounting Feature

Results for the getConsolidatedExchangeRate operation can include values for the book field. Results can also be filtered by this field.

- If the Multi-Book Accounting feature is enabled:
  - If the book is not set in the request, getConsolidatedExchangeRate returns values for all accounting books that have the Enable Consolidation option selected.
  - If the book is set in the request, getConsolidatedExchangeRate returns values for the specified accounting book.

- If the Multi-Book Accounting feature is not enabled:
  - If the book is set in the request, getConsolidatedExchangeRate returns an error with the code = "FEATURE_DISABLED" and the message = "The "Multi-Book Accounting" feature is not enabled in your NetSuite account."
  - Book is not part of the response.

**Note:** In endpoints prior to 2016.2, the getConsolidatedExchangeRate operation can only be used to return results for the primary accounting book, even if the Multi-Book Accounting feature is enabled.

Request

The GetConsolidatedExchangeRateRequest type is used for the request.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>consolidatedExchangeRateFilter</td>
<td>ConsolidatedExchangeRateFilter</td>
<td>You can filter the returned consolidated exchange rates using this filter.</td>
</tr>
</tbody>
</table>

ConsolidatedExchangeRateFilter

ConsolidatedExchangeRateFilter is defined in the core XSD.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>period</td>
<td>RecordRef</td>
<td>References an existing period. This argument is required.</td>
</tr>
</tbody>
</table>
### getConsolidatedExchangeRate

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>fromSubsidiary</td>
<td>RecordRef</td>
<td>References the receiving subsidiary. This argument is optional.</td>
</tr>
<tr>
<td>toSubsidiary</td>
<td>RecordRef</td>
<td>References the originating subsidiary. This argument is optional.</td>
</tr>
<tr>
<td>book</td>
<td>RecordRef</td>
<td>References the specific accounting book for accounts with the Multi-Book Accounting feature enabled.</td>
</tr>
</tbody>
</table>

### Response

The GetConsolidatedExchangeRateResult type is used for the response.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this operation. All applicable errors or warnings are listed within this type.</td>
</tr>
<tr>
<td>consolidatedExchangeRateList</td>
<td>ConsolidatedExchangeRateList</td>
<td>Returns a list of available consolidated exchange rates.</td>
</tr>
</tbody>
</table>

### ConsolidatedExchangeRateList

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>consolidatedExchangeRate</td>
<td>ConsolidatedExchangeRate</td>
<td>References an existing period.</td>
</tr>
</tbody>
</table>

### ConsolidatedExchangeRate

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>period</td>
<td>RecordRef</td>
<td>References an existing period.</td>
</tr>
<tr>
<td>fromSubsidiary</td>
<td>RecordRef</td>
<td>References the receiving subsidiary.</td>
</tr>
<tr>
<td>toSubsidiary</td>
<td>RecordRef</td>
<td>References the originating subsidiary.</td>
</tr>
<tr>
<td>currentRate</td>
<td>double</td>
<td>References the current rate.</td>
</tr>
<tr>
<td>averageRate</td>
<td>double</td>
<td>References the average rate.</td>
</tr>
<tr>
<td>historicalRate</td>
<td>double</td>
<td>References the historical rate.</td>
</tr>
</tbody>
</table>

### Faults

This operation can throw one of the following faults. See [SOAP Fault Status Codes](#) for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

The getConsolidatedExchangeRate and getBudgetExchangeRate operations are used in the same way. Therefore, please refer to the Sample Code for getBudgetExchangeRate.

getCurrencyRate

Use this operation to get the exchange rate between two currencies based on a certain date. The exchange rate values you are getting are those that appear in the Exchange Rate column of the Currency Exchange Rates table, which you can view at Lists > Accounting > Currency Exchange Rates.

The role used to execute this operation must have at least View level of the Currency permission (a Lists type permission).

Note: The Multiple Currencies feature must be enabled in your account before using the getCurrencyRate operation. For information on enabling this feature, see the help topic Enabling the Multiple Currencies Feature.

Request

The GetCurrencyRateRequest type is used for the request. This method accepts the argument CurrencyRateFilter. With this filter you specify what you want returned in the results.

If you do not specify anything in the filter, the resulting CurrencyRateList will contain all currency exchange rates (all combinations) for the actual date. For example, if you specify baseCurrency only (for example, Euro), you will get all rates for Euro {USD,YEN,SING,...}.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>currencyRateFilter</td>
<td>CurrencyRateFilter</td>
<td>Filter the returned currency exchange rates using this filter.</td>
</tr>
</tbody>
</table>
CurrencyRateFilter

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseCurrency</td>
<td>RecordRef</td>
<td>References the base currency. This argument is optional.</td>
</tr>
<tr>
<td>fromCurrency</td>
<td>RecordRef</td>
<td>References the from or &quot;exchange&quot; currency. This argument is optional.</td>
</tr>
<tr>
<td>effectiveDate</td>
<td>dateTime</td>
<td>This argument is optional. If a date is not provided, the date defaults to the current date.</td>
</tr>
</tbody>
</table>

Response

The GetCurrencyRateResponse type is used for the response.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this operation. All applicable errors or warnings are listed within this type.</td>
</tr>
<tr>
<td>currencyRate</td>
<td>double</td>
<td>Returns a list of available currency exchange rates.</td>
</tr>
</tbody>
</table>

Faults

This operation can throw one of the following faults. See [SOAP Fault Status Codes](#) for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- InsufficientPermissionFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

**SOAP Request**

```xml
  <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@netsuite.com</ns2:email>
    </ns1:passport>
  </soapenv:Header>
</soapenv:Envelope>
```
Java

This sample shows how to get the actual rate for tomorrow and for USD to GBP.

```java
public void getCurrencyRate() throws Exception {
    this.login();

    CurrencyRateFilter crf = new CurrencyRateFilter();
```
```java
Calendar cal = Calendar.getInstance();
cal.add(Calendar.DAY_OF_YEAR, +1); //tomorrow
crf.setEffectiveDate(cal);

RecordRef currencyRef1 = new RecordRef();
currencyRef1.setInternalId("1"); //USD
crf.setBaseCurrency(currencyRef1);

RecordRef currencyRef2 = new RecordRef();
currencyRef2.setInternalId("2"); //GBP
crf.setFromCurrency(currencyRef2);

GetCurrencyRateResult gcrr = _port.getCurrencyRate(crf);
CurrencyRateList crl = gcrr.getCurrencyRateList();
CurrencyRate cr = crl.getCurrencyRate(0);
Double rate = cr.getExchangeRate();
System.out.println("Exchange rate is = " + rate);
```

### Note:
The `mrr()` method is a convenience method for specifying a record without the need to indicate a record type.

```java
/* Make Record Ref out of an internalId */
public static RecordRef mrr(String internalId)
{
    RecordRef toRet = new RecordRef();
toRet.setInternalId(internalId);
return toRet;
}
```

## getCustomizationId

When integrating with NetSuite through SOAP web services, in many cases, you will want to know which custom objects exist in an account. You may also want to obtain metadata for these custom objects so that your application can handle any business logic that is specific to the account.

To learn which custom objects exist in an account, use the `getCustomizationId` operation to retrieve the internalIds, externalIds, and/or scriptIds of all custom objects of a specified type. These types, enumerated in `CustomizationType`, include the following:

- `crmCustomField`
- `customList`
- `customRecordType`
- `customSegment`
- `entityCustomField`
- `itemCustomField`
- `itemOptionCustomField`
- `otherCustomField`
- transactionBodyCustomField
- transactionColumnCustomField

**Note:** The CustomizationType object is defined in the coreTypes XSD.

After the IDs are returned, you can then pass the ID value(s) to either the get or getList operation to obtain metadata for specific custom objects (see Using get and getList with getCustomizationId for more details).

Note the following:

- To see the UI equivalent of a NetSuite custom object, go to Customization > Lists, Records, & Fields > [custom object type]. For information on working with each object type in SOAP web services, see the help topic SuiteBuilder Overview. For a more general understanding of the NetSuite customization model, see the help topic SuiteBuilder (Customization).
- Normally, you cannot add or update the internalId of a NetSuite object. Custom objects, however, are an exception to this rule. You can specify the ID on add, but you cannot update the ID thereafter. Also note that internalIds for custom objects can be set to any unique alphanumeric string up to 30 characters long. This string cannot include any spaces, but it can include underscores (_).
- The getCustomizationId operation also returns fields generated from custom segments. See the SOAP samples at SOAP Request for Returning Custom Segments.

### Using get and getList with getCustomizationId

If you want only the internalIds, externalIds, and/or scriptIds for custom objects of a specific type, you can call getCustomizationId on that custom type (for example, crmCustomField). However, if you also want the metadata for any or all of the custom objects referenced in the getCustomizationId response, you must invoke a get or getList request and specify each object's ID.

When using either the get or getList operations in conjunction with getCustomizationId, be aware of the following:

- If you specify more than one ID in a get | getList call, the IDs you pass are read in the following order: internalId, externalId, scriptId.
- If an invalid scriptId is specified but the internalId or externalId is correct, the scriptId is ignored.
- NetSuite will reject get | getList requests if only a scriptId is passed for record types that do not support scriptIds.

### Request

The getCustomizationIdRequest type is used for this request. It contains the following fields:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>customizationType</td>
<td>CustomizationType</td>
<td>Any of the custom object types enumerated in CustomizationType.</td>
</tr>
<tr>
<td>includeInactives</td>
<td>boolean</td>
<td>A value of true or false is required. A value of false means no inactive custom objects are returned in the response. A value of true means that both active and inactive custom objects are returned in the response.</td>
</tr>
</tbody>
</table>
To determine which of the returned custom objects are inactive, you must perform either a get or getList operation to obtain the metadata for the custom object type(s). For custom records and custom lists, the value is specified in the `isInactive` element.

Although the includeInactives argument is required, the value you set applies to custom record and custom list objects only. In NetSuite, custom fields cannot be set to active or inactive. Therefore, when using getCustomizationId on a custom field type, the value you specify for includeInactives will be ignored. Whether you specify true or false, all custom field objects (of the type specified in your request) will be returned.

### Response

The `getCustomizationIdResult` type is used for the response. It contains the following fields:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this operation. All applicable errors or warnings are listed within this type.</td>
</tr>
<tr>
<td>totalRecords</td>
<td>xsd:int</td>
<td>The total number of records for this request. Depending on the pageSize value, some or all the records may be returned in the response.</td>
</tr>
<tr>
<td>customizationRefList</td>
<td>CustomizationRef</td>
<td>A list of custom objects that correspond to the specified customization type. Also returns the internalId, externalId, and/or scriptId of each object.</td>
</tr>
</tbody>
</table>

### Faults

This operation can throw one of the following faults. See [SOAP Fault Status Codes](#) for more information on faults.

- `InvalidSessionFault`
- `InvalidCredentialsFault`
- `ExceededRequestLimitFault`
- `ExceededUsageLimitFault`
- `ExceededRecordCountFault`
- `ExceededRequestSizeFault`
- `UnexpectedErrorFault`

### Sample Code

The following sample is a two-part sample. The first part of the sample shows how to construct a `getCustomizationId` request so that the IDs for all custom record type objects in a NetSuite account are returned. Notice that in the `getCustomizationId` request, the value of the includeInactives parameter is set to false, meaning that only custom records marked as “active” in the account will be returned.
The second part of the sample shows how to take internalIds that are returned, and make a getList request to return the metadata for each custom record type.

C# Sample

```csharp
NetSuiteService nss = new NetSuiteService();

// credential code ignored here in this sample ...

// Perform getCustomizationId on custom record type
CustomizationType ct = new CustomizationType();
ct.getCustomizationTypeSpecified=true;
ct.getCustomizationType = GetCustomizationType.customRecordType;

// Retrieve active custom record type IDs. The includeInactives param is set to false.
GetCustomizationIdResult getCustIdResult = nss.getCustomizationId(ct, false);

// Retrieve the metadata of the returned custom record types
ReadResponse [] readResp = nss.getList(getCustIdResult.customizationRefList);
```

SOAP Request (getCustomizationId)

```xml
<soapenv:Body>
  <getCustomizationId xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <customizationType getCustomizationType="customRecordType"/>
    <includeInactives>false</includeInactives>
    <getCustomizationId/>
  </getCustomizationId>
</soapenv:Body>
```

SOAP Response (getCustomizationId)

```xml
<getCustomizationIdResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <platformCore:getCustomizationIdResult xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
    <platformCore:status isSuccess="true"/>
    <platformCore:totalRecords>1</platformCore:totalRecords>
    <platformCore:customizationRefList>
      <platformCore:customizationRef internalId="15" scriptId="customrecord15" type="customRecordType">
        <platformCore:name>Customer Satisfaction Survey</platformCore:name>
      </platformCore:customizationRef>
    </platformCore:customizationRefList>
  </platformCore:getCustomizationIdResult>
</getCustomizationIdResponse>
```

SOAP Request (getList)

```xml
<getList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <baseRef xmlns:q1="urn:core_2017_1.platform.webservices.netsuite.com" xsi:type="q1:CustomizationRef" internalId="15" scriptId="customrecord15" type="customRecordType">
    ...
  </baseRef>
</getList>
```
SOAP Response (getList)

```xml
<getListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <readResponseList>
        <readResponse>
            <platformCore:status isSuccess="true"
                xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            <record internalId="15" xsi:type="setupCustom:CustomRecordType"
                xmlns:setupCustom="urn:customization_2017_1.setup.webservices.netsuite.com">
                <setupCustom:recordName>Customer Satisfaction Survey</setupCustom:recordName>
                <setupCustom:includeName>true</setupCustom:includeName>
                <setupCustom:showId>true</setupCustom:showId>
                <setupCustom:showCreationDate>true</setupCustom:showCreationDate>
                <setupCustom:showCreationDateOnList>true</setupCustom:showCreationDateOnList>
                <setupCustom:isAvailableOffline>false</setupCustom:isAvailableOffline>
                <setupCustom:allowQuickSearch>false</setupCustom:allowQuickSearch>
                <setupCustom:isInactive>false</setupCustom:isInactive>
                <setupCustom:disclaimer>
                    Record numbers cannot be reverted back to names after being updated.
                </setupCustom:disclaimer>
                <setupCustom:enableNumbering>false</setupCustom:enableNumbering>
                <setupCustom:allowNumberingOverride>false</setupCustom:allowNumberingOverride>
                <setupCustom:isNumberingUpdateable>false</setupCustom:isNumberingUpdateable>
                <setupCustom:scriptId>customrecord15</setupCustom:scriptId>
                <setupCustom:fieldList>
                    <setupCustom:customField scriptId="customrecord_CONTACT_TYPE">
                        <setupCustom:label>Contact type</setupCustom:label>
                        <setupCustom:FieldType>_listRecord</setupCustom:FieldType>
                        <setupCustom:selectRecordType internalId="16"
                            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
                            <platformCore:name>Customer contact type</platformCore:name>
                        </setupCustom:selectRecordType>
                        <setupCustom:storeValue>true</setupCustom:storeValue>
                        <setupCustom:showInList>false</setupCustom:showInList>
                        <setupCustom:globalSearch>false</setupCustom:globalSearch>
                        <setupCustom:isParent>false</setupCustom:isParent>
                        <setupCustom:displayType>_normal</setupCustom:displayType>
                        <setupCustom:isMandatory>true</setupCustom:isMandatory>
                        <setupCustom:checkSpelling>false</setupCustom:checkSpelling>
                        <setupCustom:defaultChecked>false</setupCustom:defaultChecked>
                        <setupCustom:isFormula>false</setupCustom:isFormula>
                        <setupCustom:recType>15</setupCustom:recType>
                        <setupCustom:roleRestrict>false</setupCustom:roleRestrict>
                        <setupCustom:accessLevel>_edit</setupCustom:accessLevel>
                        <setupCustom:searchLevel>_edit</setupCustom:searchLevel>
                    </setupCustom:customField>
                    <setupCustom:customField scriptId="customrecord_IF_OTHER">
                        <setupCustom:label>Other contact type</setupCustom:label>
                        <setupCustom:FieldType>_freeFormText</setupCustom:FieldType>
                        <setupCustom:storeValue>true</setupCustom:storeValue>
                    </setupCustom:customField>
                </setupCustom:fieldList>
            </record>
        </readResponse>
    </readResponseList>
</getListResponse>
```
SOAP Request for Returning Custom Segments

```xml
<soapenv:Body>
  <getCustomizationId xmlns="urn:messages_2019_1.platform.webservices.netsuite.com">
    <customizationType getCustomizationType="customSegment"/>
    <includeInactives>false</includeInactives>
  </getCustomizationId>
</soapenv:Body>
```

SOAP Response with Unified Script ID Used on the Custom Segment

```xml
<soapenv:Body>
  <getCustomizationIdResponse xmlns="urn:messages_2019_1.platform.webservices.netsuite.com">
    <platformCore:getCustomizationIdResult
      xmlns:platformCore="urn:core_2019_1.platform.webservices.netsuite.com">
      <!-- Customization ID and additional details go here -->
    </platformCore:getCustomizationIdResult>
  </getCustomizationIdResponse>
</soapenv:Body>
```
Java Sample

The following Java sample show how to construct a getCustomizationId request so that the IDs for all custom record type objects are returned. In this example, inactive custom record type objects are not returned.

```java
// Perform getCustomizationId operation on the custom record type

public void getCustomizationId () throws Exception {
    CustomizationType ct = new CustomizationType();
    ct.setGetCustomizationType(GetCustomizationType.entityCustomField);
    ct.getGetCustomizationType();
    c.getCustomizationId(GetCustomizationType.customRecordType, false);
}
```

dataCenterUrls

You use the getDataCenterUrls operation to obtain the correct URL for external client access to NetSuite. getDataCenterURLs requests do not require authentication, nor any information identifying the application. The getDataCenterUrls operation ignores all authentication data, if any such data is sent.

SOAP web services versions 2019.1 and later are only available on account-specific domains. The URL for your account-specific SOAP web services domain includes your account ID. For example, if your account ID was 123456, then your account-specific SOAP web services domain would be 123456.suitetalk.api.netsuite.com. As of 2019.1, the getDataCenterUrls operation also returns your account-specific SOAP web services domain.

**Important:** Data center-specific domains include a data center identifier in the URL. Acceptance of data center-specific domains for integrations is being phased out of NetSuite. Data center-specific domains for integrations will no longer be supported for Release Preview and sandbox accounts as of 2020.1, and will no longer be supported for integrations for production accounts in 2020.2. You must transition your integrations from using hard-coded data center-specific domains in your NetSuite account. Replace hard-coded data center-specific domains with dynamic discovery methods or account-specific domains. Do not attempt to construct an account-specific domain yourself. Each account (production, Release Preview, and sandbox) has a unique account-specific domain. The correct URLs to use for each service in an account are listed on the Company URLs subtab of the Company Information page.

For more details about dynamic discovery of URLs, see Effects of Accounts’ Host Data Center on URLs and Dynamic Discovery of URLs for SOAP Web Services and RESTlet Clients.
Important: As of the 2019.1 SOAP web services endpoint, you must use account-specific domains with SOAP web services. Use the SOAP getDataCenterUrls operation to obtain the correct domain. Or, go to Setup > Company > Setup Tasks > Company Information (Administrator) in the NetSuite UI. Your domains are listed on the Company URLs subtab.

Important: If you receive the following error, either use your account-specific domain for your request for the data center URL, or use the c parameter in your request to specify your account ID. Using either your account-specific domain, or using the c parameter in your request, ensures that you receive the correct response. The routing depends on the version of your NetSuite account, and not the version of the WSDL you use.

Service Unavailable
No service is available at this URL. If you expect this service to be available for your account, ensure your account number appears in the URL as request parameter "c". Example: https://webservices.netSuite.com/services/NetSuitePort_X_V120?c=12345

In addition to the URL for SOAP web services access, the getDataCenterUrls operation also supports discovery of other NetSuite domains. You can use this operation to discover any of the following:

- restletDomain – https://<accountID>.restlets.api.netsuite.com
- systemDomain – https://<accountID>.app.netsuite.com
- webservicesDomain – https://<accountID>.suitetalk.api.netsuite.com

Note: When you send your initial dynamic discovery call, you should use the https://webservices.netsuite.com URL.

If you are building an integration for your own use or for a single customer, your integration must include logic to determine the correct URL for the appropriate NetSuite account. If you are building an application that will be distributed to multiple customers, you must incorporate logic that can determine the correct URL for any of those customers.

REST Services as Alternate Methods for Dynamic Discovery of URL

The getDataCenterUrls operation is available in the 2012.2 and later endpoints. However, you do not have to upgrade your endpoint to support dynamic discovery of the correct SOAP web services domain. NetSuite also provides REST services that you can use.

- For general information about the DataCenterUrls REST service, see DataCenterUrls REST Service.
- For general information about the REST roles service, see The REST roles Service.
- For example code that uses this REST service, see:
  - Java REST Example for SOAP Web Services Domain Discovery
  - The NetSuite Developer Resources sample applications. The sample applications incorporate RESTful code for dynamic discovery of SOAP web services domains. These samples are available at the SuiteTalk Sample Applications Page.

Request

The GetDataCenterUrlsRequest type is used for this request. It requires the following field:

SuiteTalk SOAP Web Services Platform Guide
getDataCenterUrls

Element Name | XSD Type | Notes
--- | --- | ---
account | string | NetSuite account ID.

**Note:** If an invalid account ID is specified, the default URLs are returned.

Response

The GetDataCenterUrlsResult type is used for the response. It contains the following fields:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this operation. All applicable errors or warnings are listed within this type.</td>
</tr>
<tr>
<td>restDomain</td>
<td>string</td>
<td>URL used by external client applications to access NetSuite through a RESTlet.</td>
</tr>
<tr>
<td>systemDomain</td>
<td>string</td>
<td></td>
</tr>
<tr>
<td>webservicesDomain</td>
<td>string</td>
<td>URL used by external client applications to access NetSuite through SOAP web services.</td>
</tr>
</tbody>
</table>

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- ExceededRequestSizeFault
- UnexpectedErrorFault

Also, a DATA_REQD error is thrown if no account ID is provided in the request. For an example, see SOAP Response (empty account).

Sample Code

**C# Sample**

```csharp
class DataCenterDomains : NetSuiteService
{
    private System.Uri OriginalUri;

    public DataCenterDomains (string account, bool doNotSetUrl) : base()
    {
        OriginalUri = new System.Uri(this.Url);
        if (account == null || account.Length == 0)
            account = "empty";
        if (!doNotSetUrl)
        {
            DataCenterUrls urls = getDataCenterUrls(account).dataCenterUrls;
        }
    }
```
Uri dataCenterUri = new Uri(urls.webservicesDomain + OriginalUri.PathAndQuery);
this.Url = dataCenterUri.ToString();
}

public void SetAccount(string account)
{
    if (account == null || account.Length == 0)
        account = "empty";

    this.Url = OriginalUri.AbsoluteUri;
    DataCenterUrls urls = getDataCenterUrls(account).dataCenterUrls;
    Uri dataCenterUri = new Uri(urls.webservicesDomain + OriginalUri.PathAndQuery);
    this.Url = dataCenterUri.ToString();
}

Java Sample

/**
 * Since 12.2 endpoint accounts are located in multiple data centers with different domain names.
 * To use the correct one, wrap the locator and get the correct domain first.
 * See getDataCenterUrls WSDL method documentation in the Help Center.
 */
private static class  dataCenters  extends NetSuiteServiceLocator
{
    private String account;

    public  dataCenters (String account)
    {
        this.account = account;
    }

    @Override
    public NetSuitePortType getNetSuitePort(URL defaultWsDomainURL)
    {
        try
        {
            NetSuitePortType _port = super.getNetSuitePort(defaultWsDomainURL);
            // Get the webservices domain for your account
            DataCenterUrls urls = _port.getDataCenterUrls(account).getDataCenterUrls();
            String wsDomain = urls.getWebservicesDomain();

            // Return URL appropriate for the specific account
            return super.getNetSuitePort(new URL(wsDomain.concat(defaultWsDomainURL.getPath())));
        }
        catch (Exception e)
        {
            throw new RuntimeException(e);
        }
    }
}
---

// Locate the NetSuite web service.
NetSuiteServiceLocator service = new  dataCenters ("myAccount");
// Get the service port (to the correct data center)
_port = service.getNetSuitePort(new URL("https://webservices.netsuite.com/services/NetSuitePort_2012_2"));

Java Sample

```java
public void testDataCenterUrlsPublic() throws Exception
{
    NLWsClient c = new NLWsClient();
    c.initPort();
    String account = "1326288";

    System.out.println("Account: " + account);
    DataCenterUrls urls = c.getPort().getDataCenterUrls(account).getDataCenterUrls();

    System.out.println("\tREST domain: " + urls.getRestDomain());
    System.out.println("\tWeb services domain: " + urls.getWebservicesDomain());
    System.out.println("\tSystem domain: " + urls.getSystemDomain());
    System.out.println();
}
```

SOAP Request

```xml
<?xml version="1.0" encoding="UTF-8"?>
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <getDataCenterUrls xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <account>1326288</account>
    </getDataCenterUrls>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```xml
<?xml version="1.0" encoding="utf-8"?>
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <platformMsgs:documentInfo xmlns:platformMsgs="urn:messages_2017_1.platform.webservices.netsuite.com">
      <platformMsgs:nsId>WEB SERVICES_1326288_8016201790521873260264554_10fa27f165429</platformMsgs:nsId>
    </platformMsgs:documentInfo>
  </soapenv:Header>
  <soapenv:Body>
    <getDataCenterUrlsResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <getDataCenterUrlsReturn xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
        <platformCore:status isSuccess="true"/>
      </getDataCenterUrlsReturn>
    </getDataCenterUrlsResponse>
  </soapenv:Body>
</soapenv:Envelope>
```
getDeleted

You use the getDeleted operation to retrieve a list of deleted records. When you use this operation, you can filter by record type, by script ID, and by the date of record deletion. For each record that matches the filter criteria, the system returns the following data:

- The record's type
- The record's name
- The record's internal ID
- The record's external ID (if available)
- The date the record was deleted

The getDeleted operation can be useful when you want to synchronize information in a client application with NetSuite. For example, an Outlook client application plug-in maintains a list of contacts and
synchronizes that list with NetSuite. The getDeleted operation can be used to determine contact deletions since the last synchronization with NetSuite.

Permissions

By default, only account administrators and users with the Full Access role have permission to use the getDeleted operation. However, you can grant permission to other roles. To use the getDeleted operation, the role has to have the Deleted Records permission (and the Web Services permission).

Users who have the Deleted Records permission can access results about any type of record that was deleted, even if they do not have permission to create or modify that record type. They also have permission to use the Deleted Record search type in the user interface, unless the Web Services Only option has been selected for the role.

You can give a role the Deleted Records permission by going to Setup > Users/Roles > Manage Roles. Go to the Permissions > Setup subtab, and add a line for the permission.

Usage Notes

Note the following:

- Not all record types support the getDeleted operation. For a complete list of supported record types for this operation, refer to the DeletedRecordType enumeration in the coreTypes xsd.
- Data about deleted records remains available indefinitely.
- The getDeleted response does not differentiate between records deleted through SOAP web services and through the UI.
- Even though it is not a search, the getDeleted operation respects the pageSize search preference.
- The getDeleted operation changed in 2015.1. If you are using the 2014.2 endpoint or earlier, refer to the Platform Guide for your WSDL version. To obtain an older version of the platform guide, see the help topic SOAP Web Services Archives.

Request

The GetDeletedRequest type is used for the request.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>getDeletedFilter</td>
<td>GetDeletedFilter</td>
<td>See GetDeletedFilter.</td>
</tr>
<tr>
<td>pageIndex</td>
<td>xsd: int</td>
<td>Use this argument to specify which page of results to return in your response. If there is only one page of results, enter 1.</td>
</tr>
</tbody>
</table>

GetDeletedFilter

The GetDeletedFilter lets you retrieve data based on the date, record type, and script ID of the deleted records.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>deletedDate</td>
<td>SearchDateField</td>
<td>The date that the record was deleted. For example, you can search for records that were deleted since a particular date or between two dates.</td>
</tr>
<tr>
<td>type</td>
<td>SearchEnumMultiSelectField</td>
<td>The type of the record that was deleted. For a complete list of all of the values currently supported by the getDeleted operation,</td>
</tr>
</tbody>
</table>
Element Name | XSD Type | Notes
---|---|---
scriptID | SearchStringField | The scriptID identifying the specific custom record or custom transaction type. Use this field to narrow the results to custom records (or custom transactions) only of that particular type. If you do not specify a script ID, and rely only on the type field, the operation returns details on all instances that were deleted, regardless of type.

Response

The DeletedRecord type is used for the response.

Element Name | XSD Type | Notes
---|---|---
status | Status | The status for this operation. All applicable errors or warnings are listed within this type.
totalRecords | xsd: int | The total number of records that meet the filter criteria.
pageSize | xsd: int | The total number of records listed on each page of the results.
totalPages | xsd: int | The total number of available pages.
pagIndex | xsd: int | The number identifying the current page of results.
deletedRecordList | deletedRecordList | A list of records that meet the filter criteria

Faults

This operation can throw any of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

The following sample shows how to retrieve details about deleted instances of a custom record type with the script ID customrecord1. It specifies that 10 results should be returned per page, and it includes logic to retrieve all pages.

Java

```java
{  
    c.setSearchPrefsPageSize(10);  
}
getDeleted

```java
GetDeletedFilter filter = new GetDeletedFilter();
filter.setType(new SearchEnumMultiSelectField(new String[] {DeletedRecordType._customRecord},
SearchEnumMultiSelectFieldOperator.anyOf));
filter.setScriptId(new SearchStringField("customrecord1", SearchStringFieldOperator.is));

GetDeletedResult result;
int i = 1;
do {
    result = c.m_port.getDeleted(filter, i);
    if (i == 1)
    {
        System.out.println(String.format("Page Size : %d, Total Pages : %d, Total Records : %d",
result.getPageSize(), result.getTotalPages(), result.getTotalRecords()));
    }
    processDeletedRecordList(result.getDeletedRecordList());
    System.out.println(String.format("Current Page Index : %d", result.getPageIndex()));
    i++;
} while (i <= result.getTotalPages());
```

getItemAvailability

The getItemAvailability operation can be used to retrieve the inventory availability for a specific list of items.

You can filter the returned list using a lastQtyAvailableChange filter. If set, only items with quantity available changes recorded as of this date are returned.

If the Multi-Location Inventory feature is enabled, this operation returns results for all locations. For locations that do not have any items available, only location IDs and names are listed in results.

**Warning:** This operation supports up to 10,000 records. If this limit is exceeded, an error is returned.

Request

The GetItemAvailabilityRequest type is used for the request.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>itemAvailabilityFilter</td>
<td>ItemAvailabilityFilter</td>
<td>You can filter the returned itemAvailability using this filter.</td>
</tr>
</tbody>
</table>

**ItemAvailabilityFilter**

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>RecordRefList</td>
<td>References an existing item record in NetSuite.</td>
</tr>
<tr>
<td>lastQtyAvailableChange</td>
<td>dateTime</td>
<td>If set, only items with quantity available changes recorded as of the specified date are returned.</td>
</tr>
</tbody>
</table>
Response

The GetItemAvailabilityResult type is used for the response.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this operation. All applicable errors or warnings are listed within this type.</td>
</tr>
</tbody>
</table>

### ItemAvailabilityList

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>RecordRef</td>
<td>References an existing item record.</td>
</tr>
<tr>
<td>lastQtyAvailableChange</td>
<td>dateTime</td>
<td>If set, only items with quantity available changes recorded as of this date are returned.</td>
</tr>
<tr>
<td>locationId</td>
<td>RecordRef</td>
<td>References a location in a user defined list at Lists &gt; Accounting &gt; Locations.</td>
</tr>
<tr>
<td>quantityOnHand</td>
<td>double</td>
<td>The number of units of an item in stock.</td>
</tr>
<tr>
<td>onHandValueMli</td>
<td>double</td>
<td></td>
</tr>
<tr>
<td>reorderPoint</td>
<td>double</td>
<td>The stock level at which a new order for the item needs to be placed</td>
</tr>
<tr>
<td>preferredStockLevel</td>
<td>double</td>
<td>The preferred quantity of this item maintained in inventory for a specific location.</td>
</tr>
<tr>
<td>quantityOnOrder</td>
<td>double</td>
<td>The number of units of an item pending receipt from a vendor.</td>
</tr>
<tr>
<td>quantityCommitted</td>
<td>double</td>
<td>The number of units of an item reserved by unfulfilled sales orders.</td>
</tr>
<tr>
<td>quantityBackOrdered</td>
<td>double</td>
<td>The number of units of an item reserved by unfulfilled sales orders.</td>
</tr>
<tr>
<td>quantityAvailable</td>
<td>double</td>
<td>The number of units in stock that have not been committed to fulfill sales.</td>
</tr>
</tbody>
</table>

### ItemAvailabilityFilter

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>item</td>
<td>RecordRefList</td>
<td></td>
</tr>
<tr>
<td>lastQtyAvailableChange</td>
<td>dateTime</td>
<td></td>
</tr>
</tbody>
</table>

### Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

**SOAP Request (for C# sample)**

```xml
<soap:Body>
<platformMsgs:getItemAvailability>
  <platformMsgs:itemAvailabilityFilter>
    <platformCore:item>
      <platformCore:recordRef internalId="390" type="inventoryItem">
        <platformCore:name/>
      </platformCore:recordRef>
    </platformCore:item>
    <platformCore:lastQtyAvailableChange/>
  </platformMsgs:itemAvailabilityFilter>
</platformMsgs:getItemAvailability>
</soap:Body>
```

**SOAP Response (for C# sample)**

```xml
<soapenv:Body>
<getItemAvailabilityResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <getItemAvailabilityResult xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
    <status isSuccess="true"/>
    <itemAvailabilityList>
      <itemAvailability>
        <item internalId="390" type="inventoryItem">
          <name>testItem</name>
        </item>
        <locationId internalId="1" type="location">
          <name>East Coast</name>
        </locationId>
        <quantityOnHand>20.0</quantityOnHand>
        <onHandValueMli>0.0</onHandValueMli>
        <quantityCommitted>0.0</quantityCommitted>
        <quantityAvailable>20.0</quantityAvailable>
      </itemAvailability>
      <itemAvailability>
        <item internalId="390" type="inventoryItem">
          <name>testItem</name>
        </item>
        <locationId internalId="2" type="location">
          <name>West Coast</name>
        </locationId>
      </itemAvailability>
    </itemAvailabilityList>
  </getItemAvailabilityResult>
</getItemAvailabilityResponse>
</soapenv:Body>
```
C# Sample

```csharp
private void getItemAvailability()
{
    this.login(true);

    RecordRef item1 = new RecordRef();
    item1.internalId = "59";
    item1.type = RecordType.inventoryItem;
    item1.typeSpecified = true;

    RecordRef[] recordrefs = new RecordRef[1];
    recordrefs[0] = item1;

    ItemAvailabilityFilter filter = new ItemAvailabilityFilter();
    filter.item = recordrefs;
    GetItemAvailabilityResult res = _service.getItemAvailability(filter);
}
```

Java Sample

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

    RecordRef item1 = new RecordRef();
    item1.setInternalId("25");
    item1.setType(RecordType.inventoryItem);

    RecordRef item2 = new RecordRef();
    item2.setInternalId("76");
    item2.setType(RecordType.giftCertificateItem);

    RecordRef[] recordRefArray = new RecordRef[2];
    recordRefArray[0] = item1;
    recordRefArray[1] = item2;

    RecordRefList itemRefList = new RecordRefList();
    itemRefList.setRecordRef(recordRefArray);

    ItemAvailabilityFilter itemAvailability = new ItemAvailabilityFilter();
    itemAvailability.setItem(itemRefList);

    GetItemAvailabilityResult response = _port.getItemAvailability(itemAvailability);
}
```
**getList**

The getList operation is used to retrieve a list of one or more records by providing the unique ids that identify those records.

If there are multiple ids provided, they can either belong to the same record type or different record types. For example, it is possible to retrieve a customer and a contact within a single request using this operation.

If some of the provided ids are invalid, the request is still processed for the valid ids and the response will contain a warning that indicates that some of the ids were invalid.

> **Note:** An asynchronous equivalent is available for this operation, asyncGetList. For information about asynchronous request processing, see [Synchronous Versus Asynchronous Request Processing](#).

### Request

The getListRequest type is used for the request. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>recordRef</td>
<td>RecordRef</td>
<td>An array of recordRef objects that specify the ids of the records to be retrieved.</td>
</tr>
</tbody>
</table>

### Response

The getListResponse type is used for the response. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this operation. All applicable errors or warnings are listed within this type.</td>
</tr>
<tr>
<td>recordList</td>
<td>Record[]</td>
<td>A list of records that correspond to the specified ids. The actual records returned need to be of a type that extends the abstract type Record.</td>
</tr>
</tbody>
</table>

### Faults

This operation can throw one of the following faults. See [SOAP Fault Status Codes](#) for more information on faults.

- InvalidSessionFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault
Sample Code

SOAP Request

In the following example, two records are retrieved — one customer record and one employee record. Note that you must provide the internal ID of the specify instance of the record and the record type for the getList.

```xml
<soap:Body>
  <platformMsgs:getList>
    <platformMsgs:baseRef internalId="983" type="customer" xsi:type="platformCore:RecordRef"/>
    <platformMsgs:baseRef internalId="-5" type="employee" xsi:type="platformCore:RecordRef"/>
  </platformMsgs:getList>
</soap:Body>
```

SOAP Response

```xml
<soapenv:Body>
<getListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <readResponseList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <readResponse>
      <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"/>
      <record internalId="983" xsi:type="ns2:Customer" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:ns2="urn:relationships_2017_1.lists.webservices.netsuite.com">
        <ns2:entityId>Shutter Fly</ns2:entityId>
        <ns2:isInactive>false</ns2:isInactive>
        <ns2:companyName>Shutter Fly, Inc</ns2:companyName>
        ...
        [more fields]
        ...
        <ns2:customFieldList>
          <ns6:customField internalId="265" scriptId="custentity_map"
               xsi:type="ns6:StringCustomFieldRef" xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.com">
            <ns6:value>http://maps.google.com</ns6:value>
          </ns6:customField>
        </ns2:customFieldList>
      </record>
    </readResponse>
    <readResponse>
      <ns8:status isSuccess="true" xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com"/>
      <record internalId="-5" xsi:type="ns9:Employee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:ns9="urn:employees_2017_1.lists.webservices.netsuite.com">
        <ns9:entityId>A Wolfe</ns9:entityId>
        <ns9:isInactive>false</ns9:isInactive>
        ...
        [more fields]
        ...
      </record>
    </readResponse>
  </readResponseList>
</getListResponse>
</soapenv:Body>
```
C#

```csharp
private int getCustomerList()
{
    // This operation requires a valid session
    this.login( true );

    // Prompt for list of nsKeys and put in an array
    _out.write( "nsKeys for records to retrieved (separated by commas): " );
    String reqKeys = _out.readLine();
    string[] nsKeys = reqKeys.Split( new Char[] {','} );

    return getCustomerList( nsKeys, false );
}
```

Java

```java
public int getCustomerList() throws RemoteException, ExceededUsageLimitFault, UnexpectedErrorFault, InvalidSessionFault, ExceededRecordCountFault {
    // This operation requires a valid session
    this.login(true);

    // Prompt for list of nsKeys and put in an array
    _console
        .write("nsKeys for records to retrieved (separated by commas): ");
    String reqKeys = _console.readLine();
    String[] nsKeys = reqKeys.split(",");

    return getCustomerList(nsKeys, false);
}
```

getPostingTransactionSummary

The getPostingTransactionSummary operation lets you retrieve a summary of the actual data that posted to the general ledger in an account. You can use available filters and fields to generate reports that are similar to what you see when you run financial reports such as a Trial Balance, Balance Sheet, or an Income Statement.

**Note:** For information about NetSuite financial reports and financial statements, see the following topics:

- Financial Reports
- Financial Statements Overview

The getPostingTransactionSummary operation returns the fields defined in PostingTransactionSummary. You can query by any filter defined in PostingTransactionSummaryFilter and group the results by any field defined in PostingTransactionSummaryField.
The first call to the operation returns the first page, total number of hits (totalRecords), and the number of pages. You can then retrieve subsequent pages by giving the page number.

**Note:** NetSuite caches the results after the first call to getPostingTransactionSummary as subsequent pages are being retrieved. The cache is reset if the session expires, or if you make another call to this operation with a page index of 1.

Also note the following:

- This operation can only be executed in a role that has the Financial Statements permission assigned. To enable this permission for a role, a NetSuite administrator must go to Setup > User Roles > Manage Roles, click the Reports tab, select **Financial Statements** from the Permission list, and then click Save.

- To search for null, specify -1. If the return is null, the element will be skipped (not -1).

- A maximum of 10,000 expressions is supported. A request with more than this number of expressions returns an error.

- For large reports (for example, you have chosen all the columns), the query will take a long time on first request, but subsequent requests will be fast. Make sure your timeout limit is set high.

**Important:** The information in the following paragraph pertains to NetSuite OneWorld accounts only.

The amounts returned from getPostingTransactionSummary are in the currency of the subsidiary, not the parent. If you want the amounts in the currency of the parent, you must programmatically apply the appropriate exchange rate. To obtain exchange rates, you must call getConsolidatedExchangeRate, which reads data from the Consolidated Exchange Rate table. In your code, you must then multiply amount values returned by getPostingTransactionSummary by the exchange rate values returned by getConsolidatedExchangeRate.

Although the NetSuite UI automatically consolidates all amounts, you must perform your own exchange rate calculations in SOAP web services.

**Note:** You may want to do planning in your local currency, in which case there is no need for exchange rate conversions.

### Request

The GetPostingTransactionSummaryRequest type is used for the request.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>fields</td>
<td>PostingTransactionSummaryField</td>
<td>Specify how you want your data grouped.</td>
</tr>
<tr>
<td>filters</td>
<td>PostingTransactionSummaryFilter</td>
<td>Specify your filtering criteria.</td>
</tr>
<tr>
<td>pageIndex</td>
<td>xsd:int</td>
<td>Specify the page to be returned.</td>
</tr>
</tbody>
</table>

### PostingTransactionSummaryField

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>period</td>
<td>RecordRef</td>
<td>Specify to group data by period.</td>
</tr>
<tr>
<td>account</td>
<td>RecordRef</td>
<td>Specify to group data by account.</td>
</tr>
</tbody>
</table>
### Element Name | XSD Type | Notes
--- | --- | ---
parentItem | RecordRef | Specify to group data by parent item.
item | RecordRef | Specify to group data by item.
customer | RecordRef | Specify to group data by customer.
department | RecordRef | Specify to group data by department.
class | RecordRef | Specify to group data by class.
location | RecordRef | Specify to group data by location.
subsidiary | RecordRef | Specify to group data by subsidiary.
book | RecordRef | If you use the Multi-Book Accounting feature, specify to group data by accounting book.

### PostingTransactionSummaryFilter

| Element Name | XSD Type | Notes |
--- | --- | ---
period | RecordRef | Filter your request by period.
account | RecordRef | Filter your request by account.
parentItem | RecordRef | Filter your request by parent item.
item | RecordRef | Filter your request by item.
customer | RecordRef | Filter your request by customer.
department | RecordRef | Filter your request by department.
class | RecordRef | Filter your request by class.
location | RecordRef | Filter your request by location.
subsidiary | RecordRef | Filter your request by subsidiary.
book | RecordRef | If you use the Multi-Book Accounting feature, filter your request by accounting book.

### Response

The GetPostingTransactionSummaryResult type is used for the response.

| Element Name | XSD Type | Notes |
--- | --- | ---
status | Status | The status for this operation. All applicable errors or warnings are listed within this type.
totalRecords | xsd:int | The total number of records for this search. Depending on the pageSize value, some or all the records may be returned in this response.
pagesize | xsd:int | The page size for this search.
**Element Name** | **XSD Type** | **Notes**
--- | --- | ---
totalPages | xsd:int | The total number of pages that are part of this search.
pageIndex | xsd:int | The page index for the current set of results.
postingTransactionSummaryList | PostingTransactionSummaryList | Returns a list of available transaction summary results based on filters defined in your request.

### PostingTransactionSummaryList

**Element Name** | **XSD Type** | **Notes**
--- | --- | ---
postingTransactionSummary | PostingTransactionSummary | Returns a list of available transactions that are filtered by and grouped by values specified in PostingTransactionSummaryField and PostingTransactionSummaryFilter.

### PostingTransactionSummary

**Element Name** | **XSD Type** | **Notes**
--- | --- | ---
period | RecordRef | Returns a summary based on period.
account | RecordRef | Returns a summary based on account.
parentItem | RecordRef | Returns a summary based on parent item.
item | RecordRef | Returns a summary based on item.
customer | RecordRef | Returns a summary based on customer.
department | RecordRef | Returns a summary based on department.
class | RecordRef | Returns a summary based on class.
location | RecordRef | Returns a summary based on location.
subsidiary | RecordRef | Returns a summary based on subsidiary.
book | RecordRef | If you use the Multi-Book Accounting feature, specify to group data by accounting book.
amount | double | Returns a summary based on amount.

**Important:** This argument is required.

### Faults

This operation can throw one of the following faults. See [SOAP Fault Status Codes](#) for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
Sample Code

Example 1

The sample provided shows how to get and print account values using the getPostingTransactionSummary operation.

Java

```java
/* Make Record Ref out of an internalId */
public static RecordRef mrr(String internalId)
{
    RecordRef toRet = new RecordRef();
    toRet.setInternalId(internalId);
    return toRet;
}

public void testPostingActivity() throws Exception
{
    //c.setHttpPort(80);
    c.login();
    PostingTransactionSummaryField pagb = new PostingTransactionSummaryField();
    pagb.setParentItem(Boolean.TRUE);
    //pagb.set_class(Boolean.FALSE);
    //pagb.setItem(Boolean.FALSE);
    //pagb.setCustomer(Boolean.FALSE);
    //pagb.setLocation(Boolean.FALSE);
    PostingTransactionSummaryFilter paf = new PostingTransactionSummaryFilter();
    //paf.setAccount(new RecordRefList(new RecordRef[]{mrr("5"), mrr("12")}));
    long start = System.currentTimeMillis();
    GetPostingTransactionSummaryResult res = c.getPort().getPostingTransactionSummary(pagb, paf, 1);
    for (int i=0; i<10; i++)
    {
        if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getAccount() != null)
            System.out.println("Account:" +
            res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getAccount().getInternalId());
        if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getPeriod() != null)
            System.out.println("Period:" +
            res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getPeriod().getInternalId());
        if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getCustomer() != null)
            System.out.println("Customer:" +
            res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getCustomer().getInternalId());
    }
}
```
if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).get_class() != null)
    System.out.println("_class:" +
            res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).get_class().getInternalId());
if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getDepartment() != null)
    System.out.println("Department:" +
            res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getDepartment().getInternalId());
if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getLocation() != null)
    System.out.println("Location:" +
            res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getLocation().getInternalId());
if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getSubsidiary() != null)
    System.out.println("Subsidiary:" +
            res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getSubsidiary().getInternalId());
if (res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getItem() != null)
    System.out.println("Item:" +
            res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getItem().getInternalId());
    System.out.println("Amount:" +
            res.getPostingTransactionSummaryList().getPostingTransactionSummary(i).getAmount());
}
for (int i=res.getPageIndex(); i<res.getTotalPages(); i++)
{
    System.err.println("Elapsed Time : " + (System.currentTimeMillis() - start));
    res = c.getPort().getPostingTransactionSummary(pagb, paf, i+1);
    System.err.println("Elapsed Time : " + (System.currentTimeMillis() - start));
    GetPostingTransactionSummaryResult p = c.getPort().getPostingTransactionSummary(pagb, paf,
    res.getTotalPages()+2);
    assertFalse(p.getStatus().isIsSuccess());
}

Example 2
The following SOAP and Java show how to get the posting activity for posting period 109. The sample
prints the unconsolidated and consolidated amounts in the parent subsidiary by child subsidiary and
account – for example, if subsidiary 2 is in Euros and subsidiary 1, the parent company, is in dollars, show
the amount in Euros and the amount in dollars for that account in period 109.

SOAP Request
<soapenv:Body>
    <getPostingTransactionSummary xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
        <fields>
            <ns6:period xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.com">true</ns6:period>
            <ns7:account xmlns:ns7="urn:core_2017_1.platform.webservices.netsuite.com">true</ns7:account>
            <ns8:subsidiary xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com">true</ns8:subsidiary>
        </fields>
        <filters>
            <ns9:period xmlns:ns9="urn:core_2017_1.platform.webservices.netsuite.com">
                <ns9:recordRef internalId="109"/>
            </ns9:period>
        </filters>
        <pageIndex>1</pageIndex>
    </getPostingTransactionSummary>
</soapenv:Body>
SOAP Response

```xml
<soapenv:Body>
    <getPostingTransactionSummaryResponse
        xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
        <platformCore:getPostingTransactionSummaryResult
            xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
            <platformCore:status isSuccess="true"/>
            <platformCore:totalRecords>19</platformCore:totalRecords>
            <platformCore:pageSize>1000</platformCore:pageSize>
            <platformCore:totalPages>0</platformCore:totalPages>
            <platformCore:pageIndex>1</platformCore:pageIndex>
            <platformCore:postingTransactionSummaryList>
                <platformCore:postingTransactionSummary>
                    <platformCore:period internalId="109"/>
                    <platformCore:account internalId="2"/>
                    <platformCore:subsidiary internalId="1"/>
                    <platformCore:amount>500.0</platformCore:amount>
                </platformCore:postingTransactionSummary>
                <platformCore:postingTransactionSummary>
                    <platformCore:period internalId="109"/>
                    <platformCore:account internalId="6"/>
                    <platformCore:subsidiary internalId="1"/>
                    <platformCore:amount>19.0</platformCore:amount>
                </platformCore:postingTransactionSummary>
                <platformCore:postingTransactionSummary>
                    <platformCore:period internalId="109"/>
                    <platformCore:account internalId="6"/>
                    <platformCore:subsidiary internalId="2"/>
                    <platformCore:amount>5.0</platformCore:amount>
                </platformCore:postingTransactionSummary>
                <platformCore:postingTransactionSummary>
                    <platformCore:period internalId="109"/>
                    <platformCore:account internalId="6"/>
                    <platformCore:subsidiary internalId="4"/>
                    <platformCore:amount>6.0</platformCore:amount>
                </platformCore:postingTransactionSummary>
                <platformCore:postingTransactionSummary>
                    <platformCore:period internalId="109"/>
                    <platformCore:account internalId="16"/>
                    <platformCore:subsidiary internalId="6"/>
                    <platformCore:amount>-4.0</platformCore:amount>
                </platformCore:postingTransactionSummary>
                <platformCore:postingTransactionSummary>
                    <platformCore:period internalId="109"/>
                    <platformCore:account internalId="24"/>
                    <platformCore:subsidiary internalId="8"/>
                    <platformCore:amount>-100.0</platformCore:amount>
                </platformCore:postingTransactionSummary>
            </platformCore:postingTransactionSummaryList>
        </platformCore:getPostingTransactionSummaryResult>
    </getPostingTransactionSummaryResponse>
</soapenv:Body>
```
<platformCore:postingTransactionSummary>
  <platformCore:period internalId="109"/>
  <platformCore:account internalId="58"/>
  <platformCore:subsidiary internalId="8"/>
  <platformCore:amount>100.0</platformCore:amount>
</platformCore:postingTransactionSummary>

<platformCore:postingTransactionSummary>
  <platformCore:period internalId="109"/>
  <platformCore:account internalId="112"/>
  <platformCore:subsidiary internalId="1"/>
  <platformCore:amount>-496.0</platformCore:amount>
</platformCore:postingTransactionSummary>

<platformCore:postingTransactionSummary>
  <platformCore:period internalId="109"/>
  <platformCore:account internalId="112"/>
  <platformCore:subsidiary internalId="5"/>
  <platformCore:amount>-1.0</platformCore:amount>
</platformCore:postingTransactionSummary>

<platformCore:postingTransactionSummary>
  <platformCore:period internalId="109"/>
  <platformCore:account internalId="139"/>
  <platformCore:subsidiary internalId="4"/>
  <platformCore:amount>-9.0</platformCore:amount>
</platformCore:postingTransactionSummary>

<platformCore:postingTransactionSummary>
  <platformCore:period internalId="109"/>
  <platformCore:account internalId="142"/>
  <platformCore:subsidiary internalId="8"/>
  <platformCore:amount>3.0</platformCore:amount>
</platformCore:postingTransactionSummary>

<platformCore:postingTransactionSummary>
  <platformCore:period internalId="109"/>
  <platformCore:account internalId="145"/>
  <platformCore:subsidiary internalId="5"/>
  <platformCore:amount>1.0</platformCore:amount>
</platformCore:postingTransactionSummary>

<platformCore:postingTransactionSummary>
  <platformCore:period internalId="109"/>
  <platformCore:account internalId="148"/>
  <platformCore:subsidiary internalId="2"/>
  <platformCore:amount>-5.0</platformCore:amount>
</platformCore:postingTransactionSummary>

<platformCore:postingTransactionSummary>
  <platformCore:period internalId="109"/>
  <platformCore:account internalId="153"/>
  <platformCore:subsidiary internalId="6"/>
  <platformCore:amount>4.0</platformCore:amount>
</platformCore:postingTransactionSummary>

<platformCore:postingTransactionSummary>
  <platformCore:period internalId="109"/>
  <platformCore:account internalId="155"/>
  <platformCore:subsidiary internalId="1"/>
  <platformCore:amount>-23.0</platformCore:amount>
</platformCore:postingTransactionSummary>

<platformCore:postingTransactionSummary>
Java

/* Make Record Ref out of an internalId */
public static RecordRef mrr(String internalId)
{
    RecordRef toRet = new RecordRef();
    toRet.setInternalId(internalId);
    return toRet;
}

public void testPostingWorkflow() throws Exception
{
    c.setCredentials(CRED_DB96_SIC);
    c.useRequestLevelCredentials();

    // Show and group by subsidiary, period and account.
    // These are basic columns. If you do not include account,
    // all amounts will be 0.
    PostingTransactionSummaryField pagb = new PostingTransactionSummaryField();
    pagb.setSubsidiary(Boolean.TRUE);
    pagb.setPeriod(Boolean.TRUE);
    pagb.setAccount(Boolean.TRUE);

    PostingTransactionSummaryFilter paf = new PostingTransactionSummaryFilter();
    paf.setPeriod(new RecordRefList(new RecordRef[]{mrr("109")}));

    GetPostingTransactionSummaryResult gestalt =
        c.getPort().getPostingTransactionSummary(pagb,paf,1);

    ConsolidatedExchangeRateFilter f = new ConsolidatedExchangeRateFilter();
    f.setPeriod(mrr("109"));
    f.setToSubsidiary(mrr("1"));

    GetConsolidatedExchangeRateResult cerr =
        c.getPort().getConsolidatedExchangeRate(f);

    for (PostingTransactionSummary unConsolidated :

getPostingTransactionSummary

```java
getalt.getPostingTransactionSummaryList().getPostingTransactionSummary()
{
    ConsolidatedExchangeRate rate = null;
    for (ConsolidatedExchangeRate testRate :
        cerr.getConsolidatedExchangeRateList().getConsolidatedExchangeRate())
    {
        if
            (testRate.getFromSubsidiary().getInternalId().equals
                (unConsolidated.getSubsidiary().getInternalId()))
        {
            rate = testRate;
            break;
        }
    }
    if (rate == null)
        continue;  // the Target Subsidiary will not have a conversion rate to itself.
    System.out.println("For Subsidiary: " + unConsolidated.getSubsidiary().getInternalId() + " to Consolidated Parent (1).\nPeriod: " + unConsolidated.getPeriod().getInternalId() + "\nAccount Id: " + unConsolidated.getAccount().getInternalId() + "\nUnconsolidated Amount is: " + unConsolidated.getAmount() + "\nConsolidated Amounts are (Avg/Historical/Current): " + (unConsolidated.getAmount()*rate.getAverageRate()) + " / " + (unConsolidated.getAmount()*rate.getHistoricalRate()) + " / " + (unConsolidated.getAmount()*rate.getCurrentRate()));
}
```

getSavedSearch

This operation allows users to retrieve a list of existing saved search IDs on a per-record-type basis (for example, all saved search IDs for every Customer saved search). Note that after you retrieve the list of saved search IDs, you may need to look in the NetSuite UI to see the criteria defined for the saved search. To navigate to the list of saved searches in the NetSuite UI, go to Lists > Search > Saved Searches.

This API takes a search record type as a request argument and returns a list of record references of the saved search. For each saved search, the response includes the internalId, scriptId, and name (if you are using the 2013.2 WSDL or later — with previous WSDLs, the scriptId is not returned).

For use cases explaining why you would want to get a list of saved search IDs and then reference a specific ID in your code, see Reference Existing Saved Searches.

**Note:** There is no async equivalent for this operation.

**Request**

The GetSavedSearchRequest type is used for the request. It contains the following fields:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>GetSavedSearchRecord</td>
<td></td>
</tr>
</tbody>
</table>
Response

The GetSavedSearchResponse type is used for the response. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this operation. All applicable errors or warnings are listed within this type.</td>
</tr>
<tr>
<td>totalRecords</td>
<td>xsd:int</td>
<td>The total number of records for this search. Depending on the pageSize value, some or all the records may be returned in this response</td>
</tr>
<tr>
<td>recordRefList</td>
<td>Record[]</td>
<td>A list of records that correspond to the specified ids. The actual records returned need to be of a type that extends the abstract type Record.</td>
</tr>
</tbody>
</table>

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

C#}

```csharp
GetSavedSearchRecord record = new GetSavedSearchRecord();
record.searchTypeSpecified = true;
record.searchType = SearchRecordType.transaction;

_service.getSavedSearch(record);
```

Java

```java
public void getSavedSearches() throws RemoteException{
  this.login(true);
  GetSavedSearchRecord record = new GetSavedSearchRecord();
  record.setSearchType(SearchRecordType.transaction);
}```
getSavedSearchResult result = _port.getSavedSearch(record);
}

SOAP Request

<soap:Body>
  <getSavedSearch xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <record searchType="transaction"/>
  </getSavedSearch>
</soap:Body>

SOAP Response

<soapenv:Body>
  <getSavedSearchResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <platformCore:getSavedSearchResult xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
      <platformCore:status isSuccess="true"/>
      <platformCore:totalRecords>5</platformCore:totalRecords>
      <platformCore:recordRefList>
        <platformCore:recordRef xsi:type="platformCore:CustomizationRef" internalId="26" scriptId="customsearch26">
          <platformCore:name>Custom Transaction Search</platformCore:name>
        </platformCore:recordRef>
        <platformCore:recordRef xsi:type="platformCore:CustomizationRef" internalId="27" scriptId="customsearch27">
          <platformCore:name>CC Transaction Search</platformCore:name>
        </platformCore:recordRef>
        <platformCore:recordRef xsi:type="platformCore:CustomizationRef" internalId="42" scriptId="customsearch42">
          <platformCore:name>Returned Items</platformCore:name>
        </platformCore:recordRef>
        <platformCore:recordRef xsi:type="platformCore:CustomizationRef" internalId="44" scriptId="customsearch44">
          <platformCore:name>Ordered More than One Item</platformCore:name>
        </platformCore:recordRef>
        <platformCore:recordRef xsi:type="platformCore:CustomizationRef" internalId="46" scriptId="customsearch46">
          <platformCore:name>Spending Account Average</platformCore:name>
        </platformCore:recordRef>
      </platformCore:recordRefList>
    </platformCore:getSavedSearchResult>
  </getSavedSearchResponse>
</soapenv:Body>

getSelectValue

Use the getSelectValue operation to retrieve valid select options for a particular RecordRef, CustomRecordRef, or enumerated static field. This is useful if you are writing an application UI that needs to mimic NetSuite UI logic, if the referenced record type is not yet exposed in SOAP web services, or when
the logged-in user's role does not have permission to the instances of the referenced record type. A call to getSelectValue may return different results for the same field for different roles.

The getSelectValue operation can be used on standard body fields and custom body fields. It can also be used on sublist fields that appear on both standard and custom records.

These topics describe each aspect of the operation:

- GetSelectValue Overview
- Paginating Select Values
- Filtering Select Value Text
- Getting Dependent Select Values
- Sample Code

**Important:** If you reference a field or a select value that is renamed in future versions of NetSuite, your requests will still be handled, however, a warning will be returned. Also, when working with this operation be aware of any special permissions applied to a field. For example, a permission error will be thrown if you attempt to get select values on a field that has been disabled on a form.

**Note:** The getSelectValue operation will not return the following values: "", (blank), -1, -New-, -2, -Custom-.

GetSelectValue Overview

The following figure shows a RecordRef field (Company) as displayed in the user interface, along with a list of select values for this field. You can use getSelectValue to return the entire list of values or a subset of values.

In accounts where there are levels of field dependencies, such as OneWorld accounts, you can use getSelectValue to get select values for field "B" based on the value of field "A". The following screenshot shows an employee record in a OneWorld account. The select values associated with the Department field are based on the value specified in the Subsidiary field. In this scenario, you can use getSelectValue to get the values on the Department field by specifying the internal ID of its filterBy ("master") field, which is Subsidiary (internal ID — subsidiary). For more information, see Getting Dependent Select Values.
Running getSelectValue against records that are created from other records may result in an
INSUFFICIENT_PERMISSION error.

Paginating Select Values

The first call to getSelectValue returns the total number of select values for the specified field. If you
choose, you can return a subset of those values by specifying a pageIndex number in the request.

```xml
<complexType name="getSelectValueRequest">
  <sequence>
    <element name="fieldDescription" type="platformCore:GetSelectValueFieldDescription" minOccurs="1" maxOccurs="1"/>
    <element name="pageIndex" type="xsd:int" minOccurs="1" maxOccurs="1"/>
  </sequence>
</complexType>
```

**Note:** To define a page size, set the pageSize element in the SearchPreference type. The value
must be greater than 10 and less than the system-defined maximum of 1,000. If the number of
select values exceeds the page size, the remaining results must be retrieved in subsequent calls
getSelectValue with a new pageIndex value.

Filtering Select Value Text

To help end users pick from a long list of select values, you can use the contains, startsWith,
or is search operators to filter the results returned by getSelectValue. The filter element in the
GetSelectValueFieldDescription object lets you set the operator type in your request.

```xml
<complexType name="GetSelectValueFieldDescription">
  <sequence>
    <element name="recordType" type="platformCore:RecordType" minOccurs="0" maxOccurs="1"/>
    <element name="customRecordType" type="platformCore:RecordRef" minOccurs="0" maxOccurs="1"/>
    <element name="sublist" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <element name="field" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <element name="customForm" type="platformCore:RecordRef" minOccurs="0" maxOccurs="1"/>
    <element name="filter" type="platformCore:GetSelectValueFilter" minOccurs="0" maxOccurs="1"/>
    <element name="filterByValueList" type="platformCore:GetSelectFilterByFieldValueList" minOccurs="0" maxOccurs="1"/>
  </sequence>
</complexType>
```
Getting Dependent Select Values

The getSelectValue operation can also be used to get select values that are available on the condition of other field values. For example, in OneWorld accounts the values that appear in many dropdown fields are based on the values specified for either Customer or Subsidiary. As another example, on the Item sublist of a Sales Order, the values for the Tax Code field depend on both customer and item values.

In your getSelectValue call, you will use the `GetSelectFilterByFieldValueList` and `GetSelectFilterByFieldValue` objects to specify the filterBy field of a dependent field. (The dependent field is the field you want to get the values for. The filterBy field is the field that controls which values are available for the dependent field.)

```xml
<complexType name="GetSelectFilterByFieldValueList">
  <sequence>
    <element name="filterBy" type="platformCore:GetSelectFilterByFieldValue" minOccurs="1" maxOccurs="unbounded"/>
  </sequence>
</complexType>
<complexType name="GetSelectFilterByFieldValue">
  <sequence>
    <element name="sublist" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    <element name="field" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    <element name="internalId" type="xsd:string" minOccurs="1" maxOccurs="1"/>
  </sequence>
</complexType>
```

The sublist, field, and internalId arguments will contain:
- the name of the sublist the filterBy field appears on (if it appears on a sublist)
- the schema name of the filterBy field (for example, "entity")
- the internalId of the specific entity record (for example, "87")

A getSelectValue call for a dependent field with no filterBy ("master") field specified returns 0 records. This type of call also returns a warning that notes the filterBy field, so the call can be corrected.

**Important:** Currently there is no programmatic way to discover what the filterBy field is for another field. Note, however, on transaction records the Customer (entity) field is always the filterBy field for any dependent RecordRef field. For all other record types, you must use the UI to see which field is the filterBy field.

Request

The getSelectValueRequest type is used for the request. It contains the following elements.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>fieldDescription</td>
<td>GetSelectValueFieldDescription</td>
<td>Use to specify all characteristics of the field containing the select values. For example, depending on the field and the values you want returned, you will specify the names or internalIds for the record type, sublist, and field. You may also specify filtering criteria to refine the select options returned in the response.</td>
</tr>
</tbody>
</table>
### GetSelectValueFieldDescription

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageIndex</td>
<td>xsd: int</td>
<td>For select values that span multiple pages, use this argument to specify which page to return in your response.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>recordType</td>
<td>RecordType</td>
<td>Specify a record defined in <code>coreTypes.xsd</code></td>
</tr>
<tr>
<td>customRecordType</td>
<td>RecordRef</td>
<td>If you are getting select values for a field appearing on a custom record, specify the internal or external ID of the custom record, as well as the custom record type.</td>
</tr>
<tr>
<td>sublist</td>
<td>xsd:string</td>
<td>If getting select values for a field on a sublist, you must specify the sublist name (as it is defined in the schema). Sublist names are those names that appear toward the bottom of a record schema and are appended with <code>List</code>, for example <code>itemList</code>, <code>saleTeamList</code>, <code>timeList</code>.</td>
</tr>
<tr>
<td>field</td>
<td>xsd:string</td>
<td>Specify a field name, as defined in the schema for that record. The field value can represent either a body field or a sublist field.</td>
</tr>
<tr>
<td>customForm</td>
<td>RecordRef</td>
<td>If the RecordRef or CustomRecordRef field is associated with a custom form, specify the internal or external ID of the custom form, as well as the custom form type.</td>
</tr>
<tr>
<td>filter</td>
<td>GetSelectValueFilter</td>
<td>If you choose, you can filter select options using the <code>contains</code>, <code>is</code>, or <code>startsWith</code> operator. Use any of these operators to return a subset of the options associated with the RecordRef or CustomRecordRef field. For example, to get a specific list of customers on an opportunity record, you can search for “Adam” with the <code>contains</code> operator to get only customers whose name contains Adam. See also Filtering Select Value Text.</td>
</tr>
<tr>
<td>filterByValueList</td>
<td>GetSelectFilterByFieldValueList</td>
<td>This will contain a reference to the filterBy field (or fields) in which you specify the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>sublist</code> – If the filterBy field is on a sublist, you will specify the name of the sublist containing this field. Sublist names are appended with <code>List</code> (for example <code>itemList</code>).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>field</code> – The name of the filterBy field (for example, <code>entity</code> or <code>subsidiary</code>).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- <code>internalId</code> – The internalId value of the filterBy field (for example, “87” or “112”). If the filterBy field on an Opportunity record is Customer, you would specify: <code>sublist = null</code> <code>field = &quot;entity&quot;</code> <code>internalId = &quot;87&quot;</code> (87 being the internalId of a specific customer record, such as the Abe Simpson customer record)</td>
</tr>
</tbody>
</table>

See also Getting Dependent Select Values.
Response

The getSelectValueResponse type is used for the response. It contains the following elements.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this operation. All applicable errors or warnings are listed within this type.</td>
</tr>
<tr>
<td>totalRecords</td>
<td>xsd:int</td>
<td>The total number of record references for this search. Depending on the pageSize value, some or all the references may be returned in the response.</td>
</tr>
<tr>
<td>totalPages</td>
<td>xsd:int</td>
<td>The total number of pages for this search. Depending on the pageSize value, some or all the pages may be returned in this response.</td>
</tr>
<tr>
<td>baseRefList</td>
<td>BaseRef[]</td>
<td>An array of baseRefs that references existing NetSuite records, including custom records. These baseRefs represent the valid values available for the current field.</td>
</tr>
</tbody>
</table>

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

### SOAP Request

```xml
  <soapenv:Header>
    <ns1:passport soapenv:actor="http://schemas.xmlsoap.org/soap/actor/next" soapenv:mustUnderstand="0">
      <ns1:email xmlns:ns1="urn:messages_2017_1.platform.webservices.netsuite.com">kwolfe@netsuite.com</ns1:email>
      <ns1:password xmlns:ns1="urn:messages_2017_1.platform.webservices.netsuite.com">mypassword</ns1:password>
    </ns1:passport>
  </soapenv:Header>
  <soapenv:Body>
    <getSelectValue xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <!-- contents of getSelectValue request go here -->
    </getSelectValue>
  </soapenv:Body>
</soapenv:Envelope>
```
Java

This sample shows how to get select values for the Item field that appears on the Item sublist of a Sales Order record.

In this sample the variable “c” is a shortcut for the application that needs to communicate with NetSuite via SOAP Web Services. For more information on building an application, see SOAP Web Services Quick Start.

```java
GetSelectFilterByFieldValueList myFilterByList = new GetSelectFilterByFieldValueList(  
GetSelectFilterByFieldValue[]{new GetSelectFilterByFieldValue(null,"entity","8")});

GetSelectValueFieldDescription myGSVField = new GetSelectValueFieldDescription(RecordType.salesOrder,  
null, "itemList", "item", null, null, myFilterByList);

BaseRef[] br = c.getSelectValue(myGSVField);
```

getServerTime

This operation takes no arguments and returns the NetSuite server time in GMT, regardless of a user's time zone. Developers do not have to rely on client time when writing synchronization procedures because the client time may not be in sync with the NetSuite server time.

If you choose, you can write client code that takes the GMT returned by NetSuite and convert the time to your local time zone. The format of the dateTime value returned by getServerTime contains integer-valued year, month, day, hour and minute properties, a decimal-valued second property and a boolean time-zoned property (timestamp) – for example, 2005-09-21T15:24:00.000-07:00, where 2005-09-21 is the date, 15:24:00.000 is the time and -07:00 is your own time zone offset, if you choose to set one.

Any user with an active session can call getServerTime. There are no explicit permissions for this operation. For example, if you need to synchronize customer data with NetSuite, you can call getServerTime to initialize your synchronization process. To synchronize new or changed customers records, you can search customers and filter by lastModifiedDate using the value returned in getServerTime. To synchronize deleted customer records, you can call the getDeleted operation and use the value returned in getServerTime. This makes your synchronization process independent of your client time, which may not be in sync with the NetSuite server time and hence the timestamp we use to track changes.
Request
The GetServerTimeRequest type is used for the request. It takes no arguments.

Response
The GetServerTimeResponse type is used for the response. It contains the following elements.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>GetServerTimeResult</td>
<td>See GetServerTimeResult, below.</td>
</tr>
</tbody>
</table>

GetServerTimeResult
The GetServerTimeResult type contains the following elements.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this request. All applicable errors or warnings will be listed within this type, which is defined in the platformCore XSD.</td>
</tr>
<tr>
<td>serverTime</td>
<td>dateTime</td>
<td>dateTime value returned is in GMT.</td>
</tr>
</tbody>
</table>

Faults
This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.
- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- UnexpectedErrorFault

Sample Code

Java

```java
public void testGetServerTime() throws Exception {
    c.login();

    GetServerTimeResult rs = c.getPort().getServerTime();
    System.out.println("Welcome to NetSuite. At the sound of the tone the NetSuite time will be: "+
            new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(rs.getServerTime().getTime()));

    System.out.println("This compares with a client time of "+
            new SimpleDateFormat("yyyy-MM-dd HH:mm:ss").format(Calendar.getInstance().getTime()));
    System.out.println("This represents a skew of "+
            (Calendar.getInstance().getTimeInMillis() - rs.getServerTime().getTimeInMillis())/1000 + " seconds from netsuite
            (where positive time means the client clock is comparatively fast.");
```
initialize / initializeList

Use the initialize operation to emulate the UI workflow by prepopulating fields on transaction line items with values from a related record. Your SOAP web services application can then modify only the values it needs to before submitting the record.

For example, in the UI clicking Bill from a Sales Order record loads an Invoice record where fields are populated with values from the Sales Order. When loading an invoice record in SOAP web services, you can reference the related Sales Order record to initialize fields with values from that sales order.

**Note:** An asynchronous equivalent is available for the initializeList operation, `asyncInitializeList`. For information about asynchronous request processing, see [Synchronous Versus Asynchronous Request Processing](#).

The following table lists all the transaction types that can be used with the initialize operation and the valid reference types they can use.

<table>
<thead>
<tr>
<th>(Target Record) Transaction Type</th>
<th>Initialize Reference</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembly Unbuild</td>
<td>Assembly Build</td>
<td></td>
</tr>
<tr>
<td>Cash Refund</td>
<td>Cash Sale</td>
<td></td>
</tr>
<tr>
<td>Cash Refund</td>
<td>Return Authorization</td>
<td></td>
</tr>
<tr>
<td>Cash Sale</td>
<td>Customer</td>
<td>Populates Billable tabs</td>
</tr>
<tr>
<td>Cash Sale</td>
<td>Estimate</td>
<td></td>
</tr>
<tr>
<td>Cash Sale</td>
<td>Opportunity</td>
<td></td>
</tr>
<tr>
<td>Cash Sale</td>
<td>Sales Order</td>
<td></td>
</tr>
<tr>
<td>Credit Memo</td>
<td>Customer</td>
<td></td>
</tr>
<tr>
<td>Credit Memo</td>
<td>Invoice</td>
<td></td>
</tr>
<tr>
<td>Credit Memo</td>
<td>Return Authorization</td>
<td></td>
</tr>
<tr>
<td>Customer Deposit</td>
<td>Sales Order</td>
<td></td>
</tr>
<tr>
<td>Customer Payment</td>
<td>Customer</td>
<td>Use the arAccount parameter to specify the accounts receivable (AR) account in a Customer to Customer Payment initialization. (This parameter is define in InitializeAuxRefType.)</td>
</tr>
<tr>
<td>Customer Payment</td>
<td>Invoice</td>
<td></td>
</tr>
<tr>
<td>Customer Refund</td>
<td>Credit Memo</td>
<td></td>
</tr>
<tr>
<td>Customer Refund</td>
<td>Customer</td>
<td></td>
</tr>
<tr>
<td>Deposit Application</td>
<td>Customer Deposit</td>
<td></td>
</tr>
<tr>
<td>Estimate</td>
<td>Opportunity</td>
<td></td>
</tr>
<tr>
<td>Invoice</td>
<td>Customer</td>
<td>Defaults billable time, expense, and items</td>
</tr>
<tr>
<td>Invoice</td>
<td>Estimate</td>
<td></td>
</tr>
<tr>
<td>(Target Record) Transaction Type</td>
<td>Initialize Reference</td>
<td>Notes</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------------</td>
<td>-------</td>
</tr>
<tr>
<td>Invoice</td>
<td>Opportunity</td>
<td></td>
</tr>
<tr>
<td>Invoice</td>
<td>Sales Order</td>
<td>Defaults the line items as well as billable time, expense, and items</td>
</tr>
<tr>
<td>Item Fulfillment</td>
<td>Intercompany</td>
<td></td>
</tr>
<tr>
<td>Item Fulfillment</td>
<td>Transfer Order</td>
<td></td>
</tr>
<tr>
<td>Item Fulfillment</td>
<td>Sales Order</td>
<td>Defaults the line items for fulfillment</td>
</tr>
<tr>
<td>Item Fulfillment</td>
<td>Transfer Order</td>
<td>The line field on a transfer order does not have to correspond to the orderLine field on a newly created item fulfillment.</td>
</tr>
<tr>
<td>Item Fulfillment</td>
<td>Vendor Return</td>
<td></td>
</tr>
<tr>
<td>Item Receipt</td>
<td>Intercompany</td>
<td></td>
</tr>
<tr>
<td>Item Receipt</td>
<td>Transfer Order</td>
<td></td>
</tr>
<tr>
<td>Item Receipt</td>
<td>Purchase Order</td>
<td>Initializing an item receipt from a dropship purchase order is not supported. The recommended workflow is to initialize an item fulfillment from the sales order that has the dropship item. The accounting impact will be the same and more desirable because inventory levels will not be affected.</td>
</tr>
<tr>
<td>Item Receipt</td>
<td>Return Authorization</td>
<td></td>
</tr>
<tr>
<td>Item Receipt</td>
<td>Transfer Order</td>
<td></td>
</tr>
<tr>
<td>Purchase Order</td>
<td>Requisition</td>
<td></td>
</tr>
<tr>
<td>Return Authorization</td>
<td>Cash Sale</td>
<td></td>
</tr>
<tr>
<td>Return Authorization</td>
<td>Invoice</td>
<td></td>
</tr>
<tr>
<td>Return Authorization</td>
<td>Sales Order</td>
<td></td>
</tr>
<tr>
<td>Sales Order</td>
<td>Estimate</td>
<td></td>
</tr>
<tr>
<td>Sales Order</td>
<td>Opportunity</td>
<td></td>
</tr>
<tr>
<td>Vendor Bill</td>
<td>Purchase Order</td>
<td>You can initialize or link a single purchase order to a vendor bill, or a list of purchase orders to a vendor bill. Note, however, when linking multiple purchase orders to a single vendor bill, all purchase orders must be for the same vendor. For more information, see the help topic Linking Purchase Orders to a Vendor Bill.</td>
</tr>
<tr>
<td>Vendor Bill</td>
<td>Vendor</td>
<td>Use the apAccount parameter to specify the accounts payable (AP) account in a Vendor to Vendor Bill initialization. (This parameter is define in InitializeAuxRefType.)</td>
</tr>
<tr>
<td>Vendor Credit</td>
<td>Vendor</td>
<td></td>
</tr>
<tr>
<td>Vendor Credit</td>
<td>Vendor Bill</td>
<td></td>
</tr>
<tr>
<td>Vendor Credit</td>
<td>Vendor Return</td>
<td></td>
</tr>
<tr>
<td>Vendor Credit</td>
<td>Authorization</td>
<td></td>
</tr>
<tr>
<td>Vendor Payment</td>
<td>Employee</td>
<td></td>
</tr>
<tr>
<td>Vendor Payment</td>
<td>Vendor</td>
<td></td>
</tr>
</tbody>
</table>
initialize / initializeList

<table>
<thead>
<tr>
<th>(Target Record) Transaction Type</th>
<th>Initialize Reference</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor Payment</td>
<td>Vendor Bill</td>
<td></td>
</tr>
<tr>
<td>Vendor Return Authorization</td>
<td>Vendor</td>
<td></td>
</tr>
<tr>
<td>Vendor Return Authorization</td>
<td>Vendor Bill</td>
<td></td>
</tr>
<tr>
<td>Work Order Close</td>
<td>Work Order</td>
<td></td>
</tr>
<tr>
<td>Work Order Completion</td>
<td>Work Order</td>
<td></td>
</tr>
<tr>
<td>Work Order Issue</td>
<td>Work Order</td>
<td></td>
</tr>
</tbody>
</table>

**Ignore Read-Only Preference**

To submit an initialized record without having to remove read-only fields populated during the initialization, set the Ignore Read-Only header preference to TRUE.

⚠️ **Important:** When this preference is set to TRUE, read-only fields are ignored during the SOAP web services request. The fields still cannot be set.

**Request**

The InitializeRequest type is used for this request. It contains the following fields:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>initializeRecord</td>
<td>InitializeRecord</td>
<td></td>
</tr>
</tbody>
</table>

The InitializeRecord type takes the following fields:

- type
- reference
- auxReference
- referenceList

**Response**

The InitializeResponse type is used for the response. It contains the following fields:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this operation. All applicable errors or warnings are listed within this type.</td>
</tr>
<tr>
<td>record</td>
<td>Record</td>
<td>The record type of the resulting initialization.</td>
</tr>
</tbody>
</table>
Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

**SOAP Request (initialize)**

```xml
<soapenv:Body>
<platformMsgs:initialize xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/
xmlns:platformCoreTyp="urn:types.core_2017_1.platform.webservices.netsuite.com"
xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"
xmlns:platformMsgs="urn:messages_2017_1.platform.webservices.netsuite.com">
<platformMsgs:initializeRecord>
<platformCore:type>invoice</platformCore:type>
<platformCore:reference internalId="1513" type="salesOrder">
<platformCore:name>1511</platformCore:name>
</platformCore:reference>
</platformMsgs:initializeRecord>
</platformMsgs:initialize>
</soapenv:Body>
```

**SOAP Request (initializeList)**

```xml
<initializeList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<initializeRecord>
<ns1:type xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com">customerPayment</ns1:type>
<ns2:reference internalId="176" type="customer">
<ns3:reference internalId="176" type="customer"/>
</ns2:reference>
</initializeRecord>
<initializeRecord>
<ns3:type xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">invoice</ns3:type>
<ns4:reference internalId="176" type="customer">
</ns4:reference>
</initializeRecord>
</initializeList>
```
Java (initializeList)

```java
InitializeRef iRef1 = new InitializeRef();
iRef1.setInternalId("176");
iRef1.setType(InitializeRefType.customer);

InitializeRecord ir1 = new InitializeRecord();
ir1.setReference(iRef1);
ir1.setType(InitializeType.customerPayment);

InitializeRecord ir2 = new InitializeRecord();
ir2.setReference(iRef1);
ir2.setType(InitializeType.invoice);

sessMgr.getPort().getNetSuitePortTypePort().initializeList(new InitializeRecord[]{ir1, ir2});
```

C# (initialize)

```csharp
private void Initialize()
{
    this.login(true);

    InitializeRef ref1 = new InitializeRef();
    ref1.type = InitializeRefType.salesOrder;

    // internal id of the sales order to be converted to cash sale
    ref1.internalId = "792";
    ref1.typeSpecified = true;

    InitializeRecord rec = new InitializeRecord();
    rec.type = InitializeType.cashSale;
    rec.reference = ref1;

    ReadResponse read1 = _service.initialize(rec);
}
```

**login**

The login operation is used to authenticate a user and start a new SOAP web services session in NetSuite. The login operation is similar to the login for the UI. This operation provides a passport that includes a username, password, account and role. On success, the NetSuite server sets a cookie and establishes a session.

When submitting SOAP web services requests, the NetSuite server can accept any SOAP document as long as it is valid against the NetSuite WSDL, and it does not matter how the SOAP document was generated. You must ensure that all cookies received from the server are managed and passed back to the server in subsequent calls. There are a total of three cookies that are initially returned in the HTTP
header of the login response, including the JSESSIONID. Make sure that you do not use the JSESSIONID cookie when invoking the login operation.

**Important:** Users can also authenticate to NetSuite by providing their user credentials in the SOAP header of each request; they do not need to invoke the login operation. With user credentials provided in each request, the need for session management and separate logins is eliminated. For more information, see Request-Level Credentials.

All available SOAP web services operations require that the user first be logged in. After successfully completed, a login creates a session that allows subsequent operations to be performed without having to log in again until the session expires or the logout operation is invoked. If the session times out, the next operation fails. SOAP web services requests initiated through a client application must have the ability to execute the login operation when a session has timed out and then submit the original request again.

**Note:** For information on session management, refer to Session Management for SOAP Web Services.

If an operation is executed before a login is performed, it fails and the InvalidSessionFault is returned. Also note that after login, only one SOAP web services request may be in progress at a time for a specific session. Any attempt to violate this will result in a SOAP fault.

NetSuite SOAP web services will use the last logged-in role when users do not specify a role in the login request and no default role is set. This makes the SOAP web services login behavior consistent with the UI login behavior. Partner applications that rely on a specific role should be careful to specify that role in the login request, otherwise their users might be logged in with a role that is not appropriate for that application.

The login operation also verifies that the specified account has the Web Services feature enabled. If the feature is not enabled in your account, the SOAP fault InvalidCredentialsFault is returned with a code of WEB_SERVICES_NOT_ENABLED. (See Enabling the SOAP Web Services Feature for steps on enabling this feature.)

**Note:** The Two-Factor Authentication feature is not compatible with web services. To use SOAP web services, you must be logged in with a role that does not require 2FA. If you want to use SOAP web services with a highly privileged role, use Token-based Authentication. For more information, see the help topic Token-based Authentication (TBA).

**Request**

The LoginRequest type is used for this request. It contains the following fields:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>passport</td>
<td>Passport</td>
<td>Contains all the required credentials including username, password, account and role to authenticate the user and create a new session.</td>
</tr>
</tbody>
</table>

The Passport type includes the following elements:

- email
- password
- account
- role

**Note:** You can confirm your accountID in the NetSuite UI. As administrator, go to Support > Customer Service > Contact Support by Phone. Your account number is displayed in a pop-up box. Also, Role is not a required parameter in the WS login. However, if you do not specify a role, the user's default role must have WS permissions.

**Response**

The LoginResponse type is used for the response. This references the SessionResponse type, which includes the status and wsRoleList elements.

The wsRoleList element returns a list of roles available for the user specified in the passport. You can then use this list of roles to execute different routines depending on available roles or to re-login with a different role.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>userID</td>
<td>RecordRef</td>
<td>References an existing user record.</td>
</tr>
<tr>
<td>role</td>
<td>RecordRef</td>
<td>References an existing role record.</td>
</tr>
<tr>
<td>isDefault</td>
<td>boolean</td>
<td>If true, the role is set as the default role for this user.</td>
</tr>
<tr>
<td>isInActive</td>
<td>boolean</td>
<td>If true, this role is currently unavailable for the current user.</td>
</tr>
<tr>
<td>isLoggedInRole</td>
<td>boolean</td>
<td>If true, the role of the logged in user is used.</td>
</tr>
</tbody>
</table>

**Retrieving userID in Axis 1.1**

Following is sample code that illustrates how you can retrieve the userID from the header when using Axis 1.1 — where `port` is the service port for a particular endpoint.

```java
String userid = getHeader(port, "sessionInfo");
public static String getHeader(NetSuitePortType port, String headerName) {
    com.netsuite.webservices.platform.NetSuiteBindingStub stub =
        (com.netsuite.webservices.platform.NetSuiteBindingStub) port;
    SOAPHeaderElement [] headers = stub.getResponseHeaders();
    for (int i=0; i< headers.length; i++)
    {
        SOAPHeaderElement header = headers[i];
        if (header.getName().equals(headerName))
        {
            Iterator childElements = header.getChildElements();
            while (childElements.hasNext())
            {
                SOAPElement el = (SOAPElement) childElements.next();
                return el.getValue();
            }
        }
    }
    return null;
}
```
Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InsufficientPermissionFault
- InvalidAccountFault
- InvalidSessionFault
- InvalidCredentialsFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In the following example a user is logged in with the Administrator role as indicated by the internalId of 3. For a list of available role IDs see Internal IDs Associated with Roles.

```xml
<soap:Body>
  <platformMsgs:login>
    <platformMsgs:passport>
      <platformCore:email>jsmith@yahoo.com</platformCore:email>
      <platformCore:password>password123</platformCore:password>
      <platformCore:account>121511</platformCore:account>
      <platformCore:role internalId="3"/>
    </platformMsgs:passport>
  </platformMsgs:login>
</soap:Body>
```

SOAP Response

```xml
<soapenv:Body>
  <loginResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <sessionResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"/>
      <ns2:wsRoleList xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
      <ns2:wsRole>
        <ns2:role internalId="3">
          <ns2:name>Administrator</ns2:name>
        </ns2:role>
        <ns2:isDefault>false</ns2:isDefault>
        <ns2:isInactive>false</ns2:isInactive>
      </ns2:wsRole>
      <ns2:wsRole>
        <ns2:role internalId="15">
          <ns2:name>Employee Center</ns2:name>
        </ns2:role>
        <ns2:isDefault>false</ns2:isDefault>
        <ns2:isInactive>false</ns2:isInactive>
      </ns2:wsRole>
    </sessionResponse>
  </loginResponse>
</soapenv:Body>
```
C#

```csharp
private void login( bool isAuto )
{
  if ( !_isAuthenticated )
  {
    // Check whether this is a forced login as part of another operation
    if ( isAuto )
      _out.writeLn( "You need to first login before invoking this operation ..." );

    // Enable client cookie management. This is required.
    _service.CookieContainer = new CookieContainer();

    // Populate Passport object with all login information
    Passport passport = new Passport();
    RecordRef role = new RecordRef();

    // Determine whether to get login information from config
    // file or prompt for it
    if ( "true".Equals( _dataCollection["promptForLogin"] ) )
    {
      _out.writeLn( "Please enter your login information: " );
      _out.write( "E-mail: ");
      passport.email = _out.readLn();
      _out.write( "Password: ");
      passport.password = _out.readLn();
      _out.write( "Role nsKey (press enter for default administrator role): ");
      role.internalId = _out.readLn();
      passport.role = role;
      _out.write( "Account: ");
      passport.account = _out.readLn();
    }
    else
    {
      passport.email = _dataCollection["login.email"]; 
      passport.password = _dataCollection["login.password"]; 

```

```xml
<ns2:wsRoleList>
  <ns2:wsRole>
    <ns2:role internalId="22">
      <ns2:name>Intranet Manager</ns2:name>
      <ns2:isDefault>false</ns2:isDefault>
      <ns2:isInactive>false</ns2:isInactive>
    </ns2:role>
  </ns2:wsRole>
  <ns2:wsRole>
    <ns2:role internalId="25">
      <ns2:name>System Administrator</ns2:name>
      <ns2:isDefault>false</ns2:isDefault>
      <ns2:isInactive>false</ns2:isInactive>
    </ns2:role>
  </ns2:wsRole>
</ns2:wsRoleList>
</sessionResponse>
</loginResponse>
</soapenv:Body>
```
```java
public void login(boolean isAuto) throws RemoteException {
    if (!_isAuthenticated) {
        // Check whether this is a forced login as part of another operation
        if (isAuto)
            _console
                .writeLn("You need to first login before invoking this operation ....");

        // Populate Passport object with all login information
        Passport passport = new Passport();
        RecordRef role = new RecordRef();

        // Determine whether to get login information from config
        // file or prompt for it
        if ("true".equals(_properties.getProperty("promptForLogin"))) {
            _console.writeln("Please enter your login information: ");
            System.out.print("E-mail: ");
            passport.setEmail(_console.readLine());
            System.out.print("Password: ");
            passport.setPassword(_console.readLine());
            System.out.print("Role nsKey (press enter for default administrator role): ");
            role.setInternalId(_console.readLine());
            role.setRole(_console.readLine());
            System.out.print("Account: ");
            passport.setAccount(_console.readLine());
        } else {
```
passport.setEmail(_properties.getProperty("login.email"));
passport.setPassword(_properties.getProperty("login.password"));
role.setInternalId(_properties.getProperty("login.roleNSkey"));
passport.setRole(role);
passport.setAccount(_properties.getProperty("login.acct"));

// Login to NetSuite
_console.info("Logging into NetSuite");
_console.info("Username: " + passport.getEmail());
_console.info("Account: " + passport.getAccount());

Status status = (_port.login(passport)).getStatus();
// Process the response
if (status.isIsSuccess() == true) {
    _isAuthenticated = true;
    _console.info("The login was successful and a new session has been created.");
} else {
    // Should never get here since any problems with the login should have resulted in a SOAP fault
    _console.error(getStatusDetails(status));
}

logout

The logout operation is used to terminate an active session.

**Note:** If you explicitly log out of a session, and then attempt to utilize the same session, the SESSION_TIMED_OUT error message is returned.

Request

The logoutRequest type is used for the request. It does not contain any fields.

Response

The status type is used for the response. It does not contain any fields.

Faults

This operation can throw one of the following faults. See [SOAP Fault Status Codes](#) for more information on faults.

- InvalidCredentialsFault
- InvalidSessionFault
- ExceededRequestLimitFault
- UnexpectedErrorFault
Sample Code

Following is an example that contains an excerpt of the SOAP body for both the request and response.

**SOAP Request**

```xml
<logout/>
```

**SOAP Response**

```xml
<logoutResponse>
  <status isSuccess="true"/>
</logoutResponse>
```

**C#**

```csharp
private void logout()
{
  if ( _isAuthenticated )
  {
    _out.info( "\nLogging out of NetSuite\n" );
    // Logout from NetSuite
    Status status = (_service.logout()).status;
    if ( status.isSuccess == true )
    {
      _isAuthenticated = false;
      _out.info( "Logout successful" );
    }
    else
    {
      // Should never get here since any problems with the
      // logout should have resulted in a SOAP fault
      _out.error( getStatusDetails( status ) );
    }
  }
  else
  {
    _out.info( "\nThe logout() operation cannot be invoked because there is no active session. " +
    "You must be first logged on before attempting to logout.\n" );
  }
}
```

**Java**

```java
public void logout() throws RemoteException {
  if (_isAuthenticated) {
    _console.info("\nLogging out of NetSuite\n");
  }
```
// Logout from NetSuite
_status = (_port.logout()).getStatus();

if (_status.isSuccess() == true) {
    _isAuthenticated = false;
    _console.info("Logout successful");
} else {
    // Should never get here since any problems with the
    // logout should have resulted in a SOAP fault
    _console.error(getStatusDetails(_status));
} else {
    _console
    .info("The logout() operation cannot be invoked because there is no active session. "+ "You must be first logged on before attempting to logout.");
}

mapSso

Single sign-on (SSO) refers to the procedure that allows a user of two or more user-authenticating Web applications to move between these applications using a browser, only presenting **authentication** information one time per session.

The mapSso operation supports the inbound single sign-on feature. This feature allows users to go directly from an external user-authenticating application to NetSuite, without having to separately log in to NetSuite. Validation is provided by the passing of an encrypted token, and user identification is provided by a mapping between external application credentials (remote company ID and remote user ID), and NetSuite credentials (email, password, account, and role used to log in to NetSuite).

**Important:** As of 2018.2, new Inbound Single Sign-on (SSO) solutions are no longer enabled for use with SOAP web services for new customers or new partners. Current partners may continue to sell their existing Inbound SSO solutions to new customers. However, it is strongly recommended that you consider other options, and use token-based authentication (TBA) if possible. For more information, see [Token-Based Authentication Details](#).

The mapSso operation streamlines the provisioning of inbound single sign-on access to NetSuite accounts by automating the mapping of credentials between an external application and NetSuite. If this operation is not used, this mapping is created for each user by requiring them to log in to NetSuite on their first single sign-on access.

This operation allows users to access NetSuite through inbound single sign-on without knowing their NetSuite password. Use of this operation is required for inbound single sign-integrations to Web store.

**Important:** This operation provides mapping, not a login to NetSuite. This operation does NOT provide the ability to log in using a single sign-on token, cannot provision a partner with the inbound single sign-on feature, or aid with public/private key exchange. Use of this operation implies that the account already has inbound single sign-on set up and has access to the associated partner ID and encryption keys needed to generate the token.

Be aware of the following:

- If a NetSuite role used for inbound single sign-on is deleted, the single sign-on mapping for any user with that role is automatically remapped to another role.
If a user has a single sign-on mapping set up with a particular role and that role is removed from the user, the mapping is deleted. You can set up a new mapping for that user with a different role.

Single sign-on mappings are not copied from a production account to a sandbox account when the sandbox is refreshed. These mappings must be recreated in the sandbox account for any users who require inbound single sign-on access to that account.

The number of mappings you can create is not limited. However, each mapping must be unique.

For more detailed information on the inbound single sign-on feature, see the help topic Inbound Single Sign-on.

Important: NetSuite hosts customer accounts in multiple data centers. For that reason, the correct URL for SOAP web services access varies depending on the data center hosting the account.

For details about dynamic discovery of URLs, see Effects of Accounts’ Host Data Center on URLs and Dynamic Discovery of URLs for SOAP Web Services and RESTlet Clients.

Request

The MapSsoRequest type is used for this request. It contains the following fields:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssoCredentials</td>
<td>SsoCredentials</td>
<td>Contains all the required credentials including username (email address), password, account, and role, to authenticate the user and create a new session.</td>
</tr>
</tbody>
</table>

The following table describes the fields in the SsoCredentials complex type. All are required.

<table>
<thead>
<tr>
<th>Field</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>email</td>
<td>This string is the email address used to log in to NetSuite.</td>
</tr>
<tr>
<td>password</td>
<td>This string is the password used to log in to NetSuite.</td>
</tr>
<tr>
<td>account</td>
<td>This string is the NetSuite account ID.</td>
</tr>
</tbody>
</table>

Note: You can confirm your account number in the NetSuite UI. As administrator, go to Support > Customer Service > Contact Support by Phone. Your account number is displayed in a pop-up box.

<table>
<thead>
<tr>
<th>Field</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>role</td>
<td>This references an existing role for the user. (The XSD type is RecordRef.)</td>
</tr>
</tbody>
</table>
| authentication token | This is a string representing the encrypted token. This token, prior to any hex-encoding, is of the form: \( \text{<companyID><space><userID><space><timestamp>} \)

Note: As spaces are used to delimit subtokens, the values for companyID and userID cannot include spaces.

For more information about the values in the authentication token, see the Inbound Single Sign-on documentation, in particular: Parameters used for both Application and Web Store Access.

<table>
<thead>
<tr>
<th>Field</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>partner ID</td>
<td>This is the affiliate ID of the integration partner, as provided to the partner by NetSuite.</td>
</tr>
</tbody>
</table>
Response

The MapSsoResponse type is used for the response. It does not contain any fields.

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidAccountFault
- InvalidCredentialsFault
- InvalidVersionFault
- ExceededRequestLimitFault
- UnexpectedErrorFault

Sample Code

SOAP Request

```xml
<soapenv:Envelope
    xmlns:xsd='http://www.w3.org/2001/XMLSchema'
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/
    xmlns:platformCore='urn:core_2017_1.platform.webservices.netsuite.com'
    xmlns:platformMsgs='urn:messages_2017_1.platform.webservices.netsuite.com'>
    <soapenv:Header/>
    <soapenv:Body>
        <mapSso xsi:type='platformMsgs:MapSsoRequest'>
            <ssoCredentials xsi:type='platformCore:SsoCredentials'>
                <email xsi:type='xsd:string'>user@example.com</email>
                <password xsi:type='xsd:string'>ExamplePassword1</password>
                <account xsi:type='xsd:string'>TSTDRV123456</account>
                <role xsi:type='platformCore:RecordRef' internalId='3'/>
                <authenticationToken xsi:type='xsd:string'>201F20C54BB7B134881CA1F04D3A9EC0C6C74FDE16F188EBAFAX41CF7372949C777574266364187AF1BD3BFA31D05F83D398690EC9525B</authenticationToken>
            </ssoCredentials>
            <partnerId xsi:type='xsd:string'>123456</partnerId>
        </mapSso>
    </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response

```xml
<?xml version="1.0" encoding="UTF-8"?>
    <platformMsgs:documentInfo/>
</soapenv:Envelope>
```
Java

```java
/* Generate a NetSuitePort */
NetSuiteServiceLocator nss = new NetSuiteServiceLocator();
NetSuitePortType myNetSuitePort = nss.getNetSuitePort();

/* Here your code needs to generate a valid Netsuite Sso token */
String mySsoToken = MySSOManager.getTokenFromIds
  ("myCompanyID","thisUsersUIDinMySystem");
RecordRef rr = new RecordRef();
rr.setid("23");

/* Setup the Credential */
SsoCredentials sc = new SsoCredentials();
sc.setAccount("TSTDRV00000");
sc.setEmail("bob@happycow.com");
sc.setPassword("fr0mCA");
sc.setRole(rr);
sc.setAuthenticationToken(mySsoToken);

/* Now initiate the mapping*/
try{
    SessionResponse sr = myNetSuitePort.mapSso(sc);
    if (!sr.getStatus().isIsSuccess())
        throw new Exception("Mapping Failed: " + sr.getStatus().getStatusDetail(0).getMessage());
}
```
search

The search operation is used to execute a search on a specific record type based on a set of criteria. You can search by defining search filter fields on the record, joined fields on an associated record, search return columns, or joined search return columns from an associated record. The results of the search can be complete records, or a select set of fields specified through search return columns.

Note that you can also use the search operation to return an existing saved search. You cannot use the search operation to retrieve state values. You must use the getAll operation to retrieve all state values in the system. The getAll operation will return all states, not the legal ones for your default country. Also note that the country and state must match on the address.

**Important:** The search preferences you set in the SearchPreferences object affect the search request and response. See SOAP Web Services Preferences for details.

Use the search operation to execute the following types of searches:

- **Basic search** — Execute a search on a record type based on search filter fields that are specific to that type. See Basic Searches in SOAP Web Services.
- **Joined search** — Execute a search on a record type based on search filter fields on an associated record type. See Joined Searches in SOAP Web Services
- **Advanced search** — Execute a search on a record type in which you specify search filter fields and/or search return columns or joined search columns. Using advanced search, you can also return an existing saved search. See Advanced Searches in SOAP Web Services.

**Note:** An asynchronous equivalent is available for the search operation, asyncSearch. For information about asynchronous request processing, see Synchronous Versus Asynchronous Request Processing.

Not all transaction types are supported by SOAP web services searches. For the list of unsupported transactions, see the help topic Transaction Search.

Also see the following sections for information on setting additional search filtering values and searching for custom fields:

- Setting Valid Search Values
- Setting the anyof, mine, or myteam Filtering Values
- Searching by lastModifiedDate
- Understanding Sorting in Advanced Search
- Search-Related Sample Code
- Searching for a Multi-select Custom Field

**Important:** To go directly to search-related code samples, see Search-Related Sample Code.

**Basic Searches in SOAP Web Services**

See the following topics to learn how to execute a basic search in NetSuite using SOAP web services:

- What is a basic search?
- Which SOAP objects are used in a basic search?
- Basic Search Code Sample
What is a basic search?

A basic search lets you search records of a specific type using the fields on that record as search filters. The following figure shows the UI equivalent of a basic Customer search. Start by going to Lists > Relationships > Customers > Search, and ensure that the Use Advanced Search box is not checked. You can specify one or more field values to use as filters for search results.

In a basic search, field criteria are the only values you set. You cannot specify search return columns. In SOAP web services, specifying search return columns is the equivalent of performing an advanced search. (See Advanced Searches in SOAP Web Services for details.)

In the example of a basic Customer search (see previous figure), the results returned include the record ID of every customer that has the Category field set to From Advertisement and the Status field set to Customer-Closed Won. Note that ALL the other data associated with these specific customer records are returned as well. As a consequence, in SOAP web services a basic search tends to increase the search response time.

Which SOAP objects are used in a basic search?

To perform a basic search in which you specify search filter criteria only, use:

1. `<Record>` Search
2. `<Record>` SearchBasic

For more details, see Basic Search Objects Explained and Basic Search Code Sample.

Basic Search Objects Explained

In SOAP web services, any record that supports search has a corresponding `<Record>` Search object, which contains a basic element. The basic element references a `<Record>` SearchBasic object, which defines all available search criteria (filter fields) specific to that record type.

The XSD snippet below shows the CustomerSearch object. The basic element references the CustomerSearchBasic object, which defines all available search criteria for the Customer record.

```xml
<complexType name="CustomerSearch">
</complexType>
```
<complexContent>
  <extension base="platformCore:SearchRecord">
    <sequence>
      <element name="basic" type="platformCommon: CustomerSearchBasic" minOccurs="0"/>
      <element name="callJoin" type="platformCommon:PhoneCallSearchBasic" minOccurs="0"/>
      <element name="campaignResponseJoin" type="platformCommon:CampaignSearchBasic" minOccurs="0"/>
    </sequence>
  </extension>
</complexContent>

Note:  <Record> Search objects reside in the same XSD as their corresponding record objects. In this case, both the Customer and CustomerSearch objects reside in the listRel XSD. Also note that the CustomerSearch object, like all <Record> Search objects, provide available search joins for that record type. For information on joined searches, see Joined Searches in SOAP Web Services.

This snippet shows the CustomerSearchBasic object. All <Record> SearchBasic objects are defined in the platformCommon XSD. This sample shows four of the available fields (accountNumber, address, addressee, addressLabel) that can be used as search criteria on the Customer record.

<complexType name="CustomerSearchBasic">
  <complexContent>
    <extension base="platformCore:SearchRecord">
      <sequence>
        <element name="accountNumber" type="platformCore:SearchStringField" minOccurs="0"/>
        <element name="address" type="platformCore:SearchStringField" minOccurs="0"/>
        <element name="addressee" type="platformCore:SearchStringField" minOccurs="0"/>
        <element name="addressLabel" type="platformCore:SearchStringField" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

For a code sample of a basic search, see Basic Search Code Sample.

Basic Search Behavior with User Event Scripts and Workflows

If the bodyFieldsOnly preference is set to false, a basic search triggers user event scripts and workflows. To avoid triggering user event scripts and workflows while using basic search, do one of the following:

- Set the bodyFieldsOnly preference to true. For information about this preference, see bodyFieldsOnly.
- Set the runServerSuiteScriptAndWorkflowTriggers preference to false. For information about this preference, see runServerSuiteScriptAndWorkflowTriggers.

Joined Searches in SOAP Web Services

See the following topics to learn how to execute a joined search in NetSuite using SOAP web services:
What is a joined search?

A joined search allows you search against a specific record type using the fields on an associated record as search filters. In the UI, you can identify which associated records provide joined filter criteria by first navigating to a record's search interface. For example, for the Customer search interface, go to Lists > Relationships > Customers > Search.

Check the Use Advanced Search box and wait for the page to update. Then scroll through the Filter dropdown list. Joined search records are indicated by the record name followed by an ellipsis (....). Search fields from any of the records listed (that are also currently exposed in SOAP web services) can be included in the search criteria.

Note: For a list of SOAP web services-supported records, see the help topic SOAP Web Services Supported Records.

Which SOAP objects are used in a joined search?

To perform a joined search in which you use search filter criteria from an associated record, use:

1. `<Record>` > `Search`
2. <Record> SearchBasic

For more details, see Joined Search Objects Explained and Joined Search Code Samples.

Joined Search Objects Explained

In SOAP web services, all search joins are listed in the <Record> Search object. For example, to find available search joins for the Contact or Employee record, see the ContactSearch and EmployeeSearch XSDs, respectively.

The snippet below shows the CustomerSearch object, which includes <xxx> Join elements. These elements reference search criteria available from other SOAP-supported records.

```xml
<complexType name="CustomerSearch">
    <complexContent>
        <extension base="platformCore:SearchRecord">
            <sequence>
                <element name="basic" type="platformCommon:CustomerSearchBasic" minOccurs="0"/>
                <element name="callJoin" type="platformCommon:PhoneCallSearchBasic" minOccurs="0"/>
                <element name="campaignResponseJoin" type="platformCommon:CampaignSearchBasic" minOccurs="0"/>
                <element name="caseJoin" type="platformCommon:SupportCaseSearchBasic" minOccurs="0"/>
                ......
            </sequence>
        </extension>
    </complexContent>
</complexType>
```

In this case, all search filter criteria from the PhoneCallSearchBasic, CampaignSearchBasic, and SupportCaseSearchBasic objects are available to the Customer record as joined search filters. Note that all <Record> SearchBasic objects in NetSuite SOAP web services are defined in the platformCommon XSD.

For a code sample of a joined search, see Joined Search Code Samples.

⚠️ Important: Only fields on SOAP-supported records can be specified as filter criteria for a joined search request. For a list of SOAP-supported records, see the help topic SOAP Web Services Supported Records.

Returning an Associated Joined List of Records

Using a combination of joined search and the internalId list on each record, you can retrieve a list of records for an associated list of records. For example, you can retrieve a list of contacts for a specific list of customers. To do this, you must first retrieve the desired list of internalIds for the record you need to retrieve by, and then submit that list in a joined search query to retrieve the associated list.

See Joined Search Code Samples for an example of a joined search which uses a list of records.

Advanced Searches in SOAP Web Services

Advanced searching in SOAP web services provides users with the ability to:

- Perform a search that references an existing saved search. See Reference Existing Saved Searches.
Perform a search that references an existing saved search, and then overrides existing search return columns with new search return columns. See Specify Search Criteria and Search Return Columns.

Perform a search that references an existing saved search, and then provides additional search filter criteria (on top of the criteria already specified in the saved search). See Specify Search Criteria and Search Return Columns.

Perform a search that specifies search criteria and search result columns. See Specify Search Criteria and Search Return Columns.

The SOAP web services API includes advanced search objects for all records that have an existing search interface. To see which objects are used to execute advanced searches, see Which SOAP objects are used in advanced search?

For advanced search code samples, see Advanced Search Code Samples.

**Important:** Note that advanced search functionality will not work in endpoints prior to 2008.2. Also note that searchMore, searchNext, and respective asynchronous operations can be used in advanced search, as well as all search preferences and pagination functionality. Also note that not all transaction types are supported by SOAP web services searches. For the list of unsupported transactions, see the help topic Transaction Search.

The following is an XSD snippet of the listRel XSD. This sample provides a high-level overview of advanced search objects and their relationship to the basic search object `<Record>Search`. In this sample, the objects Customer SearchAdvanced, Customer SearchRow, and Customer SearchRowBasic (not shown in sample) are considered to be advanced search objects in the SOAP web services API.

```xml
<complexType name="CustomerSearch">
  <complexContent>
    <extension base="platformCore:SearchRecord">
      <sequence>
        <element name="basic" type="platformCommon:CustomerSearchBasic" minOccurs="0"/>
        <element name="callJoin" type="platformCommon:PhoneCallSearchBasic" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<complexType name="CustomerSearchAdvanced">
  <complexContent>
    <extension base="platformCore:SearchRecord">
      <sequence>
        <element name="criteria" type="listRel:CustomerSearch" minOccurs="0"/>
        <element name="columns" type="listRel:CustomerSearchRow" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>

<complexType name="CustomerSearchRow">
  <complexContent>
    <extension base="platformCore:SearchRecord">
      <sequence>
        <element name="basic" type="platformCommon:CustomerSearchRowBasic" minOccurs="0"/>
        <element name="callJoin" type="platformCommon:PhoneCallSearchRowBasic" minOccurs="0"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

**Use to search by existing fields on the record of joined fields on associated records. See Basic Searches in Web Services and Joined Searches in Web Services.**

**Use to search by existing fields on the record and/or available search return columns. Can also be used to return an existing saved search. See Specify Search Criteria and Search Return Columns and Reference Existing Saved Searches.**

**Use to specify search return columns or search column joins to return in a search response. See Specify Search Criteria and Search Return Columns.**

**Note:** For additional SOAP and code samples related to searches, see Search-Related Sample Code, Joining Through Custom Fields, and Searches.

**Reference Existing Saved Searches**

Advanced search in SOAP web services lets you reference an existing saved search. Returning saved searches provides you with access to data that otherwise could not be returned in SOAP web services.
For example, advanced search lets you return saved searches that include formula filters (in the search criteria) or expressions.

When working with saved searches, two questions come to mind for SOAP web services users:

- Why reference an existing saved search?
- How do I reference an existing saved search?

### Why reference an existing saved search?

The following use cases illustrate possible scenarios for referencing a saved search:

- You want to return only a subset of data from a record (in other words, data that is specified through the saved search's return columns).
- Your integration application processes a set of records that are identified in a saved search. Periodically, users change the criteria of the search. By referencing a saved search ID, your code can reference their saved search. Developers do not have to change and re-test code every time the search criteria changes.
- You have a complex search that compiles data from many different records. You can create a saved search in the NetSuite UI, and then reference this search in SOAP web services rather than try to code the search in SOAP web services.
- You want to reference an existing saved search based on Leads, for example. You can return all the data provided in this search, and then define additional criteria for the search response. For example, you can return a Leads saved search and then provide additional criteria that returns the leads from this search created with today's date. In other words, if you reference a saved search and add any filter criteria to the search request, the additional criteria will be conjunctive with the saved search criteria.

### How do I reference an existing saved search?

First you must obtain the saved search ID. You can do so through the UI by going to Lists > Search > Saved Searches. The saved search ID appears in the ID column.

You can also use the `getSavedSearch` operation to programmatically retrieve a list of saved search IDs for a specific record type. Note that this operation does nothing more than return a list of saved search IDs on a per-record-type basis (for example, all saved search IDs for the customer record type).

You can then use the `search()` operation, along with the `<Record>` `SearchAdvanced` object to return the details of the saved search. The following is an example that shows how to instantiate the `CustomerSearchAdvanced` object and specify a `savedSearchId` to return.

```java
// create search object
CustomerSearchAdvanced customerSearch = new CustomerSearchAdvanced();

//set saved search id
customerSearch.savedSearchId="100";

// perform the search
NetSuiteService nss = new NetSuiteService();
SearchResult result = nss.search(customerSearch);
```

For more detailed samples, see Advanced Search Code Samples.
Usage Notes

- Only one saved search can be referenced as part of the search call.
- If you reference a saved search that contains search functions, you will get the results back. However, you cannot create an advanced search that uses search functions. Creating a search in SOAP web services that uses search functions is not currently supported.
- If you reference a saved search for which you do not have permission, the exception is ignored and fields have no value (are not in the SOAP response at all).
- If you reference a saved search that contains summary results, you will get the following error:
  
  We cannot return search columns for summary saved search < saved search ID >

  The following figure shows the UI equivalent of setting a summary type on a search return column. If a value is set, the saved search cannot be returned.

Specify Search Criteria and Search Return Columns

Similar to a basic search, an advanced search allows you search against a specific record type using the fields on that record as search filters. In addition, you can also specify a set of search return columns or joined columns to return.
This kind of advanced search is useful for retrieving only the record data you need rather than the contents of an entire record.

Advanced searches work well, for example, for users who may have a mobile client that needs to display only the name, phone number, and email address of their sales rep contact. Rather than returning all of the data on a contact record, users can create an advanced search that pulls only the relevant information. They are no longer required to download the entire record.

For advanced search code samples that show how to specify both search criteria and search return columns, see Advanced Search Code Samples.

What are search return columns?

The following figures show the UI equivalent of an advanced search that includes search return columns.

1. First, specify the basic search criteria on the Criteria subtab (in this example all customers that have a company name starting with the letter A).

2. Next, click the Results subtab to define the search return columns. When your search is executed, your search response will return only the company name, phone, and contact name of all customers whose company starts with the letter A.
The following figure shows the results of the search. Only the relevant data are returned.

<table>
<thead>
<tr>
<th>Customer Search: Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return To Criteria</td>
</tr>
<tr>
<td>[Image]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EDIT</th>
<th>VIEW</th>
<th>INTERNAL ID</th>
<th>COMPANY NAME</th>
<th>PHONE</th>
<th>EMAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edit</td>
<td>View</td>
<td>1765</td>
<td>Acme Inc.</td>
<td>617-355-1001</td>
<td><a href="mailto:info@acme.com">info@acme.com</a></td>
</tr>
<tr>
<td>Edit</td>
<td>View</td>
<td>1766</td>
<td>ABC Co.</td>
<td>212-555-6003</td>
<td><a href="mailto:smrtinc@abc.com">smrtinc@abc.com</a></td>
</tr>
<tr>
<td>Edit</td>
<td>View</td>
<td>1977</td>
<td>Inc.</td>
<td>318-555-2919</td>
<td><a href="mailto:jsmith@ainc.com">jsmith@ainc.com</a></td>
</tr>
</tbody>
</table>

**Which SOAP objects are used in advanced search?**

The following summarizes the objects used in different types of advanced searches. For additional details, see Advanced Search Objects Explained and Advanced Search Code Samples.

- To perform a search that specifies search filter criteria and search result columns, use:
  1. `<Record>` Search
  2. `<Record>` SearchBasic
  3. `<Record>` SearchRow
  4. `<Record>` SearchRowBasic

- To perform a search that references an existing saved search, use:
  1. `<Record>` SearchAdvanced

- To perform a search that references an existing saved search, and then overrides existing search return columns with new search return columns, use:
  1. `<Record>` SearchAdvanced
  2. `<Record>` SearchRow
  3. `<Record>` SearchRowBasic

- To perform a search that references an existing saved search, and then provides additional search filter criteria (that is in *addition* to the criteria already specified in the saved search), use:
  1. `<Record>` SearchAdvanced
  2. `<Record>` Search
  3. `<Record>` SearchBasic

**Advanced Search Objects Explained**

**The `<Record>` SearchAdvanced object contains:**

- a *criteria* element - references the `<Record>` Search object through which you specify standard search field criteria. (See Basic Searches in SOAP Web Services for details on the `<Record>` Search object.)
- a *columns* element - references the `<Record>` SearchRow object through which you specify a set of search result columns to return in the response.
- a *savedSearchId* attribute - references the saved search internal ID (for example, 57, 99, 63).
**Note:** If the Show Internal ID preference is enabled in your NetSuite account, the saved search internal ID appears in the **Internal ID** column in a saved search list. Programmatically, you can use `getSavedSearch` to obtain a list of saved search internal IDs for a specific record type.

- A *savedSearchScriptId* attribute - references a custom saved search ID (for example, customsearch_mySpecialSearch).

**Note:** If the Show Internal ID preference is enabled in your NetSuite account, the saved search ID appears in the **ID** column in a saved search list.

**Important:** You cannot use `getSavedSearch` to return a list of custom saved search IDs; only the system-defined internal IDs (57, 99, 63, etc.) will be returned.

The XSD sample below shows the CustomerSearchAdvanced object:

```xml
<complexType name="CustomerSearchAdvanced ">
  <complexContent>
    <extension base="platformCore:SearchRecord">
      <sequence>
        <element name="criteria " type="listRel:CustomerSearch" minOccurs="0"/>
        <element name="columns " type="listRel:CustomerSearchRow" minOccurs="0"/>
      </sequence>
      <attribute name="savedSearchId " type="xsd:string"/>
      <attribute name="savedSearchScriptId " type="xsd:string"/>
    </extension>
  </complexContent>
</complexType>
```

The `<Record>`SearchRow object contains:

- A *basic* element - references the `<Record>`SearchRowBasic object, which specifies available search return columns and column joins for that record type.
- `<xxx>`Join elements - references the `<Record>`SearchRowBasic object, which specifies search return columns and column joins for the associated record type.

The XSD below shows a snippet of the CustomerSearchRow object:

```xml
<complexType name="CustomerSearchRow ">
  <complexContent>
    <extension base="platformCore:SearchRow">
      <sequence>
        <element name="basic " type="platformCommon:CustomerSearchRowBasic"
          minOccurs="0"/>
        <element name="callJoin " type="platformCommon:PhoneCallSearchRowBasic"
          minOccurs="0"/>
        <element name="campaignResponseJoin " type="platformCommon:CampaignSearchRowBasic"
          minOccurs="0"/>
        <element name="caseJoin " type="platformCommon:SupportCaseSearchRowBasic"
          minOccurs="0"/>
        ...
      </sequence>
    </extension>
  </complexContent>
</complexType>
```
The `<Record>` SearchRowBasic object contains:

- search result column elements - use to define specific column names in your response.

The next snippet shows the CustomerSearchRowBasic object, which, like ALL `<Record>` SearchRowBasic objects, resides in the platformCommon XSD. This object lists all available search return columns for the Customer record.

```xml
<complexType name="CustomerSearchRowBasic">
  <sequence>
    <element name="accountNumber" type="platformCore:SearchColumnStringField" minOccurs="0" maxOccurs="unbounded"/>
    <element name="address" type="platformCore:SearchColumnStringField" minOccurs="0" maxOccurs="unbounded"/>
    <element name="addressee" type="platformCore:SearchColumnStringField" minOccurs="0" maxOccurs="unbounded"/>
    <element name="addressLabel" type="platformCore:SearchColumnStringField" minOccurs="0" maxOccurs="unbounded"/>
    ...
  </sequence>
</complexType>
```

Note that all search return columns reference SearchColumn<xxx>Field objects. See Search Column Custom XML Schema Types for definitions of each search column object.

**Important:** When executing an advanced search, you can set the SearchPreferences. `returnSearchColumns` preference to TRUE to ensure that only search return columns are returned in a search. An error is thrown if `returnSearchColumns` is set to TRUE and you have not specified search return columns in your request. (Note that you will not receive an error if you are using advanced search functionality to return a saved search that already includes search return columns.)

**Note:** Also note that in an advanced search, the `bodyFieldsOnly` preference is ignored.

The default value for `returnSearchColumns` is TRUE.

```xml
<complexType name="SearchPreferences">
  <sequence>
    <element name="bodyFieldsOnly" minOccurs="0" type="xsd:boolean" default="true"/>
    <element name="returnSearchColumns" minOccurs="0" type="xsd:boolean" default="true"/>
    <element name="pageSize" minOccurs="0" type="xsd:int"/>
  </sequence>
</complexType>
```

**How Custom Fields Are Identified in Search Results**

When executing an advanced search, you might choose to include as part of your search results values from custom fields. In some cases, the way that the system identifies these values varies depending on which endpoint you are using. For many users, this difference might not be of great consequence. However, you may want to review this information if you are upgrading from a previous endpoint to the 2013.2 WSDL or later, and if your integration depends on these values.
Specifically, this difference applies to scenarios in which you execute an advanced search, join to a record that has a custom field, and include the custom field in your search results. With 2013.1 and earlier endpoints, the system in its response adds a prefix to the string used to identify the custom field. In 2013.2 and later, this prefix is not added.

For example, suppose you had a custom item field on your inventory item record, and this custom field had an ID of custitem_style.

First, note that this value is referred to as internalId when using 2013.1 or an earlier endpoint. With 2013.2 and later endpoints, the value is called scriptId.

Second, with older endpoints, in some cases, the system modifies this value slightly. For example: Suppose you executed a TransactionSearchAdvanced and joined to the item record. If you included in your search results the custom field pictured above, the 2013.1 endpoint, or an earlier one, would identify the field’s internalId as having a value of “item_custitem_style.” That is, in this particular scenario, the system would add the “item_” prefix. With a different type of join, a different prefix would be added.

If you are using the 2013.2 endpoint or later, the system does not add a prefix. The scriptId returned in the response is the same as in the request (“custitem_style,” in this example). Therefore, if you upgrade to 2013.2 or a later WSDL and if you have integrations that depend on this value, you must modify these integrations.

**Note:** For code samples in Java and C#, see Searching for a Multi-select Custom Field.

**Example — Behavior with 2013.1 and Earlier Endpoints**

In the following example, the user executes a TransactionSearchAdvanced operation with a join to the item record. The user is searching for estimates that are associated with inventory items. In the results, the user wants to include the value of the custom field with the internalId “custitem_color.”
The response might look like the following. Note that the internalId value is now "item_custitem_color."

```xml
<platformCore:searchRowList>
    <platformCore:searchRow xsi:type="tranSales:TransactionSearchRow"
        xmlns:tranSales="urn:sales_2017_1.transactions.webservices.netsuite.com">
        <tranSales:itemJoin xmlns:platformCommon="urn:common_2017_1.platform.webservices.netsuite.com">
            <platformCommon:customFieldList>
                <platformCore:customField internalId="item_custitem_color"
                    xsi:type="platformCore:SearchColumnSelectCustomField">
                    <platformCore:searchValue internalId="1" typeId="6" />
                </platformCore:customField>
            </platformCommon:customFieldList>
        </tranSales:itemJoin>
    </platformCore:searchRow>
</platformCore:searchRowList>
```

**Example — Behavior with 2013.2 and Later Endpoints**

The SOAP request below represents another TransactionSearchAdvanced that joins to the item record. (In this example, because the 2013.2 WSDL is being used, the "custitem_color" value is identified as the scriptId, rather than internalId.)

```xml
<soapenv:Body>
    <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
        <searchRecord xsi:type="ns6:TransactionSearchAdvanced"
            xmlns:ns6="urn:sales_2017_1.transactions.webservices.netsuite.com">
            <ns6:criteria xsi:type="ns6:TransactionSearch">
                <ns6:basic xsi:type="ns7:TransactionSearchBasic"
                    xmlns:ns7="urn:common_2017_1.platform.webservices.netsuite.com">
                    <ns7:scriptId operator="anyOf" xsi:type="ns8:SearchMultiSelectField"
                        xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com">
                        <ns8:searchValue internalId="67" xsi:type="ns8:RecordRef"/>
                    </ns7:scriptId>
                </ns6:basic>
            </ns6:criteria>
        </searchRecord>
    </search>
</soapenv:Body>
```
In the SOAP response, the scriptId value – “custitem_color” – is unchanged. No prefix has been added.

Joining Through Custom Fields

If you are using 2013.2 or a later WSDL, you may notice that many RecordType Search objects include CustomSearchJoin. This object lets you join to records that are associated with the primary record through a custom field.

That is, if Record A includes a custom field of type list/record or multiple select, and this custom field references another record that is exposed to SOAP web services, the 2013.2 WSDL and later WSDLs allow your search of Record A to include a join to the SearchBasic of the referenced record.

Similarly, many RecordType SearchRow objects now include CustomSearchRowBasic. You can use this object to include columns from the referenced record in the results of an advanced search.

See the following for more details on executing searches using CustomSearchJoin and CustomSearchRowBasic:

- Overview of CustomSearchJoin and CustomSearchRowBasic
- CustomSearch Objects Defined
Overview of CustomSearchJoin and CustomSearchRowBasic

This topic describes some of the basics of using CustomSearchJoin and CustomSearchRowBasic.

Executing a Search

To use CustomSearchJoin, you can execute either a joined search or an advanced search.

For example, consider an employee record that has been altered to include a custom field pointing to the account record. In this case, you can search the employee record and use CustomSearchJoin to include data from the account record as part of your search criteria. You would execute your search using either of the following two objects:

- **EmployeeSearch** — Results returned would include the entire employee record for each employee that meets your criteria. (For general details on this type of search, see [Joined Searches in SOAP Web Services](#).)
- **EmployeeSearchAdvanced** — Results returned could include any columns you specify from the employee record or the account record. (For general details on this type of search, see [Advanced Searches in SOAP Web Services](#).) In this scenario, you would use CustomSearchRowBasic as well as CustomSearchJoin.

Sample Use Cases for CustomSearchJoin

You can use CustomSearchJoin in a couple of different scenarios.

A common case is one in which you have altered a standard record type to include a custom field that references another standard record type. This scenario includes the example where your employee record has been altered to include a custom field that points to the account record.

Additionally, you can join to and from custom record types. For example, suppose your organization uses custom record types called Customer Survey and Customer Suggestion. If Customer Survey and Customer Suggestion both include fields that reference each other, a search of Customer Survey could include a join to Customer Suggestion and vice versa.

Similarly, you can join between the following:

- A customized record type that references a standard record type.
- A standard record type that references a custom record type.

Sample Use Cases for CustomSearchRowBasic

CustomSearchRowBasic can be used in conjunction with all of the custom field usage described above for CustomSearchJoin.

Additionally, your search does not have to include the use of CustomSearchJoin in order for you to use CustomSearchRowBasic. For example, you could use EmployeeSearchAdvanced to search using criteria that is defined on the employee record (and not use any joins). But in your results, you might still want to include data from the record referenced by a custom field on the employee record. You do this using CustomSearchRowBasic.

Note that CustomSearchRowBasic is available only when you are executing an advanced search.
CustomSearch Objects Defined

For accounts that have upgraded to 2013.2 or a later endpoint, common.xsd includes two elements that support joining to and gathering result data from records that are referenced by a custom field. These objects are CustomSearchJoin and CustomSearchRowBasic. The main purpose of these objects is described in Overview of CustomSearchJoin and CustomSearchRowBasic. This topic describes how the objects are defined in the schema.

CustomSearchJoin is included in many RecordType Search objects. This element is defined as follows:

```xml
<complexType name="CustomSearchJoin">
  <sequence>
    <element name="customizationRef" type="platformCore:CustomizationRef"/>
    <element name="searchRecordBasic" type="platformCore:SearchRecordBasic"/>
  </sequence>
</complexType>
```

CustomSearchRowBasic is included in many RecordType SearchRow objects. This element is defined as follows:

```xml
<complexType name="CustomSearchRowBasic">
  <sequence>
    <element name="customizationRef" type="platformCore:CustomizationRef"/>
    <element name="searchRowBasic" type="platformCore:SearchRowBasic"/>
  </sequence>
</complexType>
```

Notice that both of these elements use CustomizationRef, which you use to identify the custom field. CustomizationRef includes five possible attributes. Of these, you can use only internal ID or script ID to identify a custom field. The other attributes (external ID, type, and name) are not applicable to custom fields.

The following example shows how you can use script ID to reference a custom field and join to AccountSearchBasic:

```java
CustomizationRef field_account = new CustomizationRef(null, null, null, null, "custentity_account");
CustomSearchJoin customSearchJoin = new CustomSearchJoin(field_account, accountSearchBasic);
```

For a more extensive example, see Example – Using CustomSearchJoin and CustomSearchRowBasic.

Note: CustomSearchJoin and CustomSearchRowBasic are available only for accounts that have upgraded to the 2013.2 or a later endpoint.

Example – Using CustomSearchJoin and CustomSearchRowBasic

If you are using the 2013.2 WSDL or later, you have access to CustomSearchJoin and CustomSearchRowBasic. This topic shows how you might use these objects in your code. For details
on the general purpose of these objects and sample use cases, see Overview of CustomSearchJoin and CustomSearchRowBasic.

The example below assumes that your employee record has been altered to include a custom field that references the account record. This sample shows how you might use EmployeeSearchAdvanced to find employee records associated with an account of a certain type and value.

```java
public  void testSearch1() throws Exception{

    // We are searching for employees associated with a certain type of account. Here we specify that account.
    AccountSearchBasic accountSearchBasic = new AccountSearchBasic();
    accountSearchBasic.setType(new SearchEnumMultiSelectField(new String[]{AccountType.__bank}, SearchEnumMultiSelectFieldOperator.anyOf));

    // Here we specify the field on the employee record that references the account record.
    CustomizationRef field_account = new CustomizationRef();
    field_account.setScriptId("custentity_account");

    // Here we create a custom search join object.
    CustomSearchJoin customSearchJoin = CustomSearchJoin(field_account, accountSearchBasic);

    // Next, we pass the custom search join that we just created into an EmployeeSearch instance.
    EmployeeSearch employeeSearch = new EmployeeSearch();
    employeeSearch.setCustomSearchJoin(new CustomSearchJoin[]{customSearchJoin});

    // Next, we specify that we want the results to include the balance of the account.
    AccountSearchRowBasic accountSearchRowBasic = new AccountSearchRowBasic();
    accountSearchRowBasic.setBalance(new SearchColumnDoubleField[]{new SearchColumnDoubleField()});
    CustomSearchRowBasic customSearchRowBasic =new CustomSearchRowBasic(field_account,accountSearchRowBasic);

    // Here, we specify that we want the employee's internal ID included in the results.
    EmployeeSearchRowBasic employeeSearchRowBasic = new EmployeeSearchRowBasic();
    employeeSearchRowBasic.setInternalId(new SearchColumnSelectField[]{new SearchColumnSelectField()});

    // In the following snippet, we pass the specified search columns into an EmployeeSearchRow instance.
    EmployeeSearchRow employeeSearchRow = new EmployeeSearchRow();
    employeeSearchRow.setCustomSearchJoin(new CustomSearchRowBasic[]{customSearchRowBasic});
    employeeSearchRow.setBasic(employeeSearchRowBasic);

    // Create the advanced search.
    EmployeeSearchAdvanced employeeSearchAdvanced = new EmployeeSearchAdvanced();
```
employeeSearchAdvanced.setCriteria(employeeSearch);
employeeSearchAdvanced.setColumns(employeeSearchRow);

c.search(employeeSearchAdvanced);

}  

Note: CustomSearchJoin and CustomSearchRowBasic are available only for accounts that have upgraded to the 2013.2 or a later endpoint.

CustomSearchJoin Usage Notes

If you are using the 2013.2 WSDL or later, you have access to CustomSearchJoin. Refer to the following sections to better understand the capabilities of this search object:

- Use CustomSearchJoin Only With Custom Fields
- CustomSearchJoin Connects Only With SearchBasics
- Nested Joins Are Not Allowed

Use CustomSearchJoin Only With Custom Fields

You cannot use CustomSearchJoin in conjunction with a standard field that references another record. For example, the standard employee record includes the Department field. If you decide to use CustomSearchJoin, and you identify the Department field as your custom field, the system returns an error reading "Invalid custom field reference." You should use the standard departmentJoin that is already provided as part of the EmployeeSearch object.

CustomSearchJoin Connects Only With SearchBasics

CustomSearchJoin allows only for a basic search of the referenced record. For example, if the employee record has been altered to include a custom field that points to the account record, you can join to AccountSearchBasic – but not AccountSearch or AccountSearchAdvanced. This limitation is consistent with the capabilities of searching in the UI.

Nested Joins Are Not Allowed

You cannot use nested joins. For example, suppose your contact record includes a custom field referencing the employee record, which includes a custom field that references the account record. A search of the contact record could include a join to the employee record, but the same search could not also join to the account record.

Setting Valid Search Values

Prior to the 2008.2 endpoint, if you performed a search that included an invalid search enum filter value, you would generally still get records returned in your search. For example, if you performed a search for all customers with the Country enum search value set to "United States" (rather than the supported enum value _unitedStates), you would still get results from your search. Although the value "United States" was
not recognized, you continued to get customer records. Note, however, the search results returned ALL customers in the system, since the value “United States” was invalid.

Starting with the 2008.2 endpoint, instead of ignoring any invalid search enum values, and still returning search results, NetSuite now returns 0 records and a no-match warning. Therefore, when setting search values, be sure to use the values defined in the schema.

The following is an example of what is now returned if invalid values are specified:

```xml
<platformCore:searchResult
  xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
  <platformCore:status isSuccess="true">
    <platformCore:statusDetail type="WARN">
      <platformCore:code>WARNING</platformCore:code>
      <platformCore:message>The field country’s enum value <United States> is invalid for this search.</platformCore:message>
    </platformCore:statusDetail>
  </platformCore:status>
  <platformCore:totalRecords>0</platformCore:totalRecords>
  <platformCore:totalPages>0</platformCore:totalPages>
  <platformCore:searchId>WEBSERVICES_MSTRWLF_10212000563721605896842316_60315faa132ad</platformCore:searchId>
  <platformCore:searchRowList/>
</platformCore:searchResult>
```

Setting the anyof, mine, or myteam Filtering Values

In SOAP web services, you can further define your search using the following filtering values:

<table>
<thead>
<tr>
<th>Filter</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>@NONE@</td>
<td>Equates to anyof (see Filtering Lists that Contain Null Values) or unassigned depending on the field.</td>
</tr>
<tr>
<td>@CURRENT@</td>
<td>Equates to mine. For example, use this filter to return all of your own events.</td>
</tr>
<tr>
<td>@HIERARCHY@</td>
<td>Equates to my team. For example, use this filter on a salesRep field for customer records. If you have previously defined the members of your sales team, using the @HIERARCHY@ filter will return only the customers that have worked with members of your sales team.</td>
</tr>
</tbody>
</table>

The following provides a SOAP sample for finding my events.

```xml
<search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <searchRecord xsi:type="ns1:CalendarEventSearchBasic"
    xmlns:ns1="urn:common_2017_1.platform.webservices.netsuite.com">
    <ns1:attendee operator="anyOf" xsi:type="ns2:SearchMultiSelectField"
      xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com">
      <ns2:searchValue internalId="@CURRENT@" xsi:type="ns2:RecordRef"/>
    </ns1:attendee>
  </searchRecord>
</search>
```

Filtering Lists that Contain Null Values

For select lists or multi selects which can have a null value, the UI supports the search criteria on these list type fields as “None Of” “-None-”, which essentially means not “Any Of” all list options. Such a search would result in all records which do NOT have this list type field as null. To accomplish this “None Of” “-None-” search you need to set the internalId of the search key to “@None@”.

SuiteTalk SOAP Web Services Platform Guide
Example

To search for Customers which have a Partner associated with them in NetSuite, the SOAP would look as below for the “Partner field not null” part:

```xml
<ns3:partner operator="noneOf">
  <ns8:searchValue internalId="@NONE@" xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com"/>
</ns3:partner>
<ns3:customFieldList>
```

Java code to generate this SOAP would be:

```java
// Adding search criteria where Partner is not null
RecordRef[] noPartner = new RecordRef[1];
noPartner[0] = new RecordRef();
noPartner[0].setInternalId("@NONE@";
SearchMultiSelectField partner = new SearchMultiSelectField();
partner.setOperator(SearchMultiSelectFieldOperator.noneOf);
partner.setSearchValue(noPartner);
custSearch.setPartner(partner);
```

C# code to generate this SOAP would be:

```csharp
// Adding search criteria where Partner is not null
RecordRef[] noPartner = new RecordRef[1];
noPartner[0] = new RecordRef();
noPartner[0].internalId = "@NONE@";
SearchMultiSelectField partner = new SearchMultiSelectField();
partner.operator = SearchMultiSelectFieldOperator.noneOf;
partner.searchValue = noPartner;
custSearch.partner = partner;
```

Searching by lastModifiedDate

This sample shows how to create a customer, and then search for the customer that was created based on a specific time frame. In this case, the sample uses the lastModifiedDate field to search “within” a couple of seconds before the customer was created and then a minute after. The search returns the record that was previously created.

Java

```java
Customer c = (Customer) new TestCustomer().createMinimalRecord();
Calendar timeFrom = Calendar.getInstance();
WriteResponse wr = sessMgr.getPort().add(c);
outputResult(wr.getStatus().isIsSuccess());
Calendar timeTo = Calendar.getInstance();
timeTo.setTimeInMillis(timeTo.getTimeInMillis() + 60000);
CustomerSearch cs = new CustomerSearch();
CustomerSearchBasic csb = new CustomerSearchBasic();
SearchDateField sdf = new SearchDateField();
```
sdf.setOperator(SearchDateFieldOperator.within);
sdf setSearchValue(timeFrom);
sdf setSearchValue2(timeTo);
csb.setLastModifiedDate(sdf);

sr = sessMgr.getWrappedPort().search(cs, this);

<searchRecord xsi:type="ns1:CustomerSearch"
xmlns:ns1="urn:relationships_2017_1.lists.webservices.netsuite.com">
  <ns1:basic xsi:type="ns2:CustomerSearchBasic"
xmlns:ns2="urn:common_2017_1.platform.webservices.netsuite.com">
    <ns2:lastModifiedDate operator="within" xsi:type="ns3:SearchDateField"
xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">
      <ns3:searchValue xsi:type="xsd:dateTime">2007-02-10T00:16:17.750Z</ns3:searchValue>
      <ns3:searchValue2 xsi:type="xsd:dateTime">2007-02-10T00:17:53.015Z</ns3:searchValue2>
    </ns2:lastModifiedDate>
  </ns1:basic>
</searchRecord>

Understanding Sorting in Advanced Search

In SOAP web services, when you return a saved search that has sorting criteria specified in the saved search, the records are returned according to the specified sorts by order. To see the sorts by criteria that has been applied to a saved search, you can look at the saved search criteria in the UI.

Note: When performing a search using SOAP web services, there is no way to specify the sort order of the returned results in the request itself.

In an ad-hoc SOAP web services search (a search in which sorts by criteria have not been set), you should be aware of the implicit sorts by order in which records are returned. This order is based on record type. Record types and the default sort by order for each record type are listed in the following table:

Note: For a list of all records associated with each type, see the help topic SOAP Web Services Supported Records.

<table>
<thead>
<tr>
<th>Record Type</th>
<th>Default “Sorted by” Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entities</td>
<td>Name (the name of the entity)</td>
</tr>
<tr>
<td>Activities</td>
<td>Event (the title of the Event)</td>
</tr>
<tr>
<td>Marketing</td>
<td>Campaign ID</td>
</tr>
</tbody>
</table>

Note: Promotion Code searches are sorted and returned by promotion code name.

<table>
<thead>
<tr>
<th>Record Type</th>
<th>Default “Sorted by” Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transactions</td>
<td>Date Created</td>
</tr>
<tr>
<td>Record Type</td>
<td>Default “Sorted by” Order</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>Support</td>
<td>Number</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Topic searches are sorted and returned by topic title.</td>
</tr>
<tr>
<td>File Cabinet</td>
<td>Name (the name of the file or folder)</td>
</tr>
<tr>
<td>Items</td>
<td>Name (the name of the item)</td>
</tr>
<tr>
<td>Communications</td>
<td>Note - sorted/returned by Author</td>
</tr>
<tr>
<td></td>
<td>Message - sorted/returned by Message Internal ID</td>
</tr>
<tr>
<td>Website</td>
<td>Name</td>
</tr>
<tr>
<td>Lists</td>
<td>Name</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Gift Certificate searches are sorted and returned by Name (From).</td>
</tr>
</tbody>
</table>

**Request**

The SearchRequest type is used for the request. It contains the following field.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchRecord</td>
<td>SearchRecord</td>
<td>The SearchRecord type is an abstract type. An instance of a type that extends SearchRecord must be used—such as CustomerSearchBasic or EventSearchBasic.</td>
</tr>
</tbody>
</table>

**Response**

The SearchResponse type is used for the response. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this search. All applicable errors or warnings will be listed within this type.</td>
</tr>
<tr>
<td>totalRecords</td>
<td>xsd:int</td>
<td>The total number of records for this search. Depending on the pageSize value, some or all the records may be returned in this response</td>
</tr>
<tr>
<td>pageSize</td>
<td>xsd:int</td>
<td>The page size for this search.</td>
</tr>
<tr>
<td>totalPages</td>
<td>xsd:int</td>
<td>The total number of pages that are part of this search.</td>
</tr>
<tr>
<td>pageIndex</td>
<td>xsd:int</td>
<td>The page index for the current set of results.</td>
</tr>
<tr>
<td>searchId</td>
<td>string</td>
<td>Returns a specific search based on its search ID.</td>
</tr>
<tr>
<td>recordList</td>
<td>Record[]</td>
<td>A list of records that meet the criteria for this search. The actual records returned need to be of a type that extends the abstract type of record.</td>
</tr>
<tr>
<td>searchRowList</td>
<td>SearchRowList</td>
<td>A list of return columns that meet the criteria for this search.</td>
</tr>
</tbody>
</table>
Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Search-Related Sample Code

See the following search-related code samples:

- Basic Search Code Sample
- Joined Search Code Samples
- Advanced Search Code Samples
- Searching for a Multi-select Custom Field

Basic Search Code Sample

SOAP Request

In the following example, customer records that have an email that contains shutterfly.com are searched for. Note that you can limit the search page size at the request level (see pageSize).

```xml
<soap:Body>
<platformMsgs:search>
<searchRecord xsi:type="ContactSearch">
  <customerJoin xsi:type="CustomerSearchBasic">
    <email operator="contains" xsi:type="platformCore:SearchStringField">
      <platformCore:searchValue>shutterfly.com</platformCore:searchValue>
    </email>
  </customerJoin>
</searchRecord>
</search>
</soap:Body>
```

SOAP Response

Notice that in this example, only one matching record was found. You can see that the page size was set such that if multiple records were found, only 10 would be returned at a time. The searchMore or searchNext operation could then be performed to return additional results.

```xml
<soapenv:Body>
<searchResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<searchResult xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
  <status isSuccess="true"/>
  <totalRecords>1</totalRecords>
  <pageSize>10</pageSize>
</searchResponse>
</soapenv:Body>
```
<totalPages>1</totalPages>
<pageIndex>1</pageIndex>
<recordList>
  <record internalId="983" xsi:type="ns1:Customer"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:ns1="urn:relationships_2017_1.lists.webservices.netsuite.com">
    <ns1:entityId>Shutter Fly</ns1:entityId>
    <ns1:isInactive>false</ns1:isInactive>
    <ns1:companyName>Shutter Fly, Inc</ns1:companyName>
    <ns1:entityStatus internalId="6"><name>LEAD-New</name>
    </ns1:entityStatus>
    <ns1:customFieldList>
      <customField internalId="165" scriptId="custentity_map" xsi:type="StringCustomFieldRef">
        <value>http://maps.google.com</value>
      </customField>
      <customField internalId="76" scriptId="custentity_had_order_problems" xsi:type="BooleanCustomFieldRef">
        <value>false</value>
      </customField>
    </ns1:customFieldList>
  </record>
</recordList>
</searchResult>
</searchResponse>
</soapenv:Body>

C#

```csharp
private void searchCustomer()
{
    // This operation requires a valid session
    this.login( true );

    _out.writeLn( "Enter search parameters" );

    // Instantiate a search object for customers. Note that the search
    // object is different from the regular record used for add and update.
    CustomerSearch custSearch = new CustomerSearch();

    // Search the customer entity id which is a string field
    _out.write( "Entity ID (press enter to skip): " );
    String nameValue = _out.readLn();
    SearchStringField entityId = null;
    if ( !nameValue.Equals( "" ) )
    {
        entityId = new SearchStringField();
        entityId.operator = SearchStringFieldOperator.contains;
        entityId.operatorSpecified = true;
        entityId.searchValue = nameValue;
        custSearch.basic.entityId = entityId; //see note below for an alternative
    }

    // Note: You could also use CustomerSearchBasic to access entityId directly, for example:
    CustomerSearchBasic custSearchBasic = new CustomerSearchBasic();
    custSearchBasic.entityId = entityId;

    // Search the customer stage which is a list field
```
String stageKeysValue = _out.readLine();
SearchMultiSelectField stage = null;
if ( !stageKeysValue.Equals( "" ) )
{
    stage = new SearchMultiSelectField();
    stage.@operator = SearchMultiSelectFieldOperator.anyOf;
    stage.operatorSpecified = true;

    string[] nskeys = stageKeysValue.Split( new Char[] {','} );

    RecordRef[] recordRefs = new RecordRef[ stageKeysValue.Length ];
    for ( int i=0; i<nskeys.Length; i++ )
    {
        RecordRef recordRef = new RecordRef();
        recordRef.internalId = nskeys[i];
        recordRefs[i] = recordRef;
    }
    stage.searchValue = recordRefs;
    custSearch.stage = stage;
}

// Search by isActive field which is a boolean
/*
SearchBooleanField isActive = new SearchBooleanField();
while ( true )
{
    _out.write( "Is active [T/F] (default is T): " );
    String upcomingStr = _out.readLine();

    if ( "T".Equals( upcomingStr.ToUpper() ) || "".Equals( upcomingStr.ToUpper() ) )
    {
        isActive.searchValue = true;
        isActive.searchValueSpecified = true;
        break;
    }
    else if ( upcomingStr.ToUpper().Equals( "F" ) )
    {
        isActive.searchValue = false;
        isActive.searchValueSpecified = true;
        break;
    }
    else
    {
        _out.writeLn( "Invalid selection" );
    }
}
custSearch.active = isActive;
*/
if ( custSearch.entityId == null && custSearch.stage == null )
{
    _out.info( "No search criteria was specified. Searching for all records." );
}
else
search

// Invoking the search() web services operation
_searchResult_ = _service.search( _custSearch_ );

// Process response
if ( _response_.status.isSuccess ) {
  // Process the records returned in the response and print to console
  _processCustomerSearchResponse_( _response_ );

  // Since pagination controls what is returned, check to see
  // if there are anymore pages to retrieve.
  _searchMore_( _response_ );
} else {
  _out_.error( _getStatusDetails( _response_.status ) );
}

Java

public void _searchCustomer_() throws _RemoteException_,
_ExceededUsageLimitFault_, _UnexpectedErrorFault_, _InvalidSessionFault_,
_ExceededRecordCountFault_ {
  // This operation requires a valid session
  _this_.login(true);

  _console_.writeLn(“Enter search parameters”);

  // Instantiate a search object for customers. Note that the search
  // object is different from the regular record used for add and update.
  _CustomerSearch_ custSearch = new _CustomerSearch();

  // Search the customer entity id which is a string field
  _console_.write(“Entity ID (press enter to skip): ”);
  _String_ nameValue = _console_.readLn();
  _SearchStringField_ entityId = null;
  if (!nameValue.equals(“”)) {
    entityId = new _SearchStringField();
    entityId.setOperator(_SearchStringFieldOperator_.contains);
    entityId.setSearchValue(nameValue);
    custSearch.setEntityId(entityId);
  }

  // Search the customer stage which is a list field
  _console_
write("Customer Stage (one or more nsKeys separated by commas, press enter to skip): ");
String stageKeysValue = _console.readLine();
SearchMultiSelectField stage = null;
if (!stageKeysValue.equals("")) {
    stage = new SearchMultiSelectField();
    stage.setOperator(SearchMultiSelectFieldOperator.anyOf);
    String[] nskeys = stageKeysValue.split(",");
    RecordRef[] recordRefs = new RecordRef[stageKeysValue.length()];
    for (int i = 0; i < nskeys.length; i++) {
        RecordRef recordRef = new RecordRef();
        recordRef.setInternalId(nskeys[i]);
        recordRefs[i] = recordRef;
    }
    stage.setSearchValue(recordRefs);
    custSearch.setStage(stage);
}
if (custSearch.getEntityId() == null && custSearch.getStage() == null) {
    _console.info("No search criteria was specified. Searching for all records.");
} else {
    _console.info("Searching for customers with the following criteria: 
" + (entityId == null ? "": 
"entityId=" + custSearch.getEntityId().getSearchValue() + 
", Operator=" + entityId.getOperator().toString()) + 
"stage nsKeys='" + stageKeysValue + ", Operator=" + stage .getOperator().toString());
}
// Set page size for number of records to be returned in search
// response
// Invoke search() web services operation
SearchResult result = _port.search(custSearch);
// Process result
if (result.getStatus().isIsSuccess()) {
    // Process the records returned in the result and print to console
    processCustomerSearchResponse(result);
    // Since pagination controls what is returned, check to see
    // if there are anymore pages to retrieve.
    searchMore(result);
} else {
    _console.error(getStatusDetails(result.getStatus()));
}
Joined Search Code Samples

Sample 1 — Customer Search With Contact Join

This sample shows how to execute a customer search in which the contact email address is specified as the search criteria.

Java

```java
public void contactSearch_with_CustomerJoin() throws Exception {
    ContactSearch cs = new ContactSearch();
    ContactSearchBasic contactSearchBasic = new ContactSearchBasic();
    contactSearchBasic.setEmail(new SearchStringField("contact@email.com", SearchStringFieldOperator.is));
    CustomerSearchBasic customerSearchBasic = new CustomerSearchBasic();
    customerSearchBasic.setEntityId(new SearchStringField("My Customer", SearchStringFieldOperator.is));
    cs.setBasic(contactSearchBasic);
    cs.setCustomerJoin(customerSearchBasic);
    sessMgr.getWrappedPort().search(cs, this);
}
```

SOAP

```xml
<search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <searchRecord xsi:type="ns4:ContactSearch"
    xmlns:ns4="urn:relationships_2017_1.lists.webservices.netsuite.com">
    <ns4:basic xsi:type="ns5:ContactSearchBasic"
      xmlns:ns5="urn:common_2017_1.platform.webservices.netsuite.com">
      <ns5:email operator="is" xsi:type="ns6:SearchStringField"
        xmlns:ns6="urn:core_2017_1.platform.webservices.netsuite.com">
        <ns6:searchValue xsi:type="xsd:string">contact@email.com</ns6:searchValue>
      </ns5:email>
    </ns4:basic>
    <ns4:customerJoin xsi:type="ns7:CustomerSearchBasic"
      xmlns:ns7="urn:common_2017_1.platform.webservices.netsuite.com">
      <ns7:entityId operator="is" xsi:type="ns8:SearchStringField"
        xmlns:ns8="urn:core_2017_1.platform.webservices.netsuite.com">
        <ns8:searchValue xsi:type="xsd:string">My Customer</ns8:searchValue>
      </ns7:entityId>
    </ns4:customerJoin>
  </searchRecord>
</search>
```

Sample 2 — Contact Search With Customer Join

The following sample shows how to return an associated joined list of records. In this case, all contacts associated with customers of internalId 1, 2 and 3 are returned.

Java

```java
RecordRef[] rr = new RecordRef[] {new RecordRef("1", RecordType.customer),
  new RecordRef("2", RecordType.customer), new RecordRef("3", RecordType.customer)};
CustomerSearchBasic customerSearchBasic = new CustomerSearchBasic();
```
customerSearchBasic.setInternalId(new SearchMultiSelectField(rr, SearchMultiSelectFieldOperator.anyOf));

ContactSearch contactSearch = new ContactSearch();
contactSearch.setCustomerJoin(customerSearchBasic);

SOAP

```xml
<soapenv:Body>
  <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <searchRecord xsi:type="ns1:ContactSearch"
      xmlns:ns1="urn:relationships_2017_1.lists.webservices.netsuite.com">
      <ns1:customerJoin xsi:type="ns2:CustomerSearchBasic"
        xmlns:ns2="urn:common_2017_1.platform.webservices.netsuite.com">
        <ns2:internalId operator="anyOf" xsi:type="ns3:SearchMultiSelectField"
          xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com">
          <ns3:searchValue internalId="1" type="customer"
            xsi:type="ns3:RecordRef"/>
          <ns3:searchValue internalId="2" type="customer"
            xsi:type="ns3:RecordRef"/>
          <ns3:searchValue internalId="3" type="customer"
            xsi:type="ns3:RecordRef"/>
        </ns2:internalId>
      </ns1:customerJoin>
    </searchRecord>
  </search>
</soapenv:Body>
```

Sample 3 — Item Search With Pricing Join

The following sample shows how to search for all items that have a price level of 10.00.

C#

```csharp
private void myItemSearch()
{
    ItemSearchBasic myItemSearchBasic = new ItemSearchBasic();

    SearchEnumMultiSelectField myEnum = new SearchEnumMultiSelectField();
    myEnum.@operator = SearchEnumMultiSelectFieldOperator.anyOf;
    myEnum.operatorSpecified = true;
    String[] searchStringArray = new String[1];
    searchStringArray[0] = "_inventoryItem";
    myEnum.operatorSpecified = true;
    myEnum.searchValue = searchStringArray;

    myItemSearchBasic.type = myEnum;

    PricingSearchBasic myPricingSearchBasic = new PricingSearchBasic();

    SearchDoubleField rateValue = new SearchDoubleField();
    rateValue.@operator = SearchDoubleFieldOperator.equalTo;
```
rateValue.operatorSpecified = true;
rateValue.searchValue = 10.00;
rateValue.searchValueSpecified = true;

myPricingSearchBasic.rate = rateValue;

ItemSearch myItemSearch = new ItemSearch();
myItemSearch.basic = myItemSearchBasic;
myItemSearch.pricingJoin = myPricingSearchBasic;

SearchResult searchResult = _service.search(myItemSearch);

SOAP Request

<soap:Body>
  <search xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <searchRecord xmlns:q1="urn:accounting_2017_1.lists.webservices.netsuite.com" xsi:type="q1:ItemSearch">
      <q1:basic>
        <type operator="anyOf" xmlns="urn:common_2017_1.platform.webservices.netsuite.com">
          <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">_inventoryItem</searchValue>
        </type>
      </q1:basic>
      <q1:pricingJoin>
        <rate operator="equalTo" xmlns="urn:common_2017_1.platform.webservices.netsuite.com">
          <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">10</searchValue>
        </rate>
      </q1:pricingJoin>
    </searchRecord>
  </search>
</soap:Body>

Advanced Search Code Samples

The following advanced search samples are provided.

1. Perform a search that includes a saved search ID.
2. Perform a search with a defined page size.
3. Perform a search in which you override search columns in a saved search.
4. Perform a search in which you specify additional saved search criteria.
5. Programmatically manipulate data returned by a search

Perform a search that includes a saved search ID.

C#

// Create a service
NetSuiteService nss = new NetSuiteService();
Perform a search with a defined page size.

This sample performs the same search as the previous sample, however, you are specifying the number of results that are returned per page.
Perform a search in which you override search columns in a saved search.

In this sample you are returning a specific saved search (57), and then overriding the search return columns in the saved search and specifying your own search return columns.

```csharp
TransactionSearchAdvanced tsa3 = new TransactionSearchAdvanced();
tsa3.savedSearchId = "57";  //substitute your own saved search internal ID

// Set search preference to return search columns
SearchPreferences sp = new SearchPreferences();
sp.pageSizeSpecified = true;
sp.pageSize = 10;
nss.searchPreferences = sp;
nss.search(tsa3);
```

```csharp
// Set search preference to return search columns
// Already defined returnSearchColumns
//SearchPreferences sp = new SearchPreferences();
//sp.returnSearchColumns = true;
//nss.searchPreferences = sp;

// Instantiate SearchColumn object
TransactionSearchRow tsr = new TransactionSearchRow();
TransactionSearchRowBasic tsrb = new TransactionSearchRowBasic();

// return internId
SearchColumnSelectField[] orderIdCols = new SearchColumnSelectField[1];
SearchColumnSelectField orderIdCol = new SearchColumnSelectField();
orderIdCol.customLabel = "Sales Order ID"; // Define a custom label
orderIdCols[0] = orderIdCol;
tsrb.internalId = orderIdCols;

SearchColumnDateField[] tranDateCols = new SearchColumnDateField[1];
SearchColumnDateField tranDateCol = new SearchColumnDateField();
tranDateCols[0] = tranDateCol;
tsrb.tranDate = tranDateCols;

SearchColumnBooleanField[] isFulfilledCols = new SearchColumnBooleanField[1];
SearchColumnBooleanField isFulfilledCol = new SearchColumnBooleanField();
isFulfilledCol.customLabel = "Order Fulfilled"; // Define a custom label
isFulfilledCols[0] = isFulfilledCol;
tsrb.shipRecvStatus = isFulfilledCols;

tsr.basic = tsrb;
tsa3.columns = tsr;
nss.search(tsa3);
```
Perform a search in which you specify additional saved search criteria.

The next sample shows how to take a saved search and specify additional search filter criteria. The additional search criteria will not override the criteria already defined in the saved search.

C#  

```csharp
TransactionSearchAdvanced tsa4 = new TransactionSearchAdvanced();
    tsa4.savedSearchId = "57";

    TransactionSearch ts = new TransactionSearch();
    TransactionSearchBasic tsb = new TransactionSearchBasic();
    // condition 1: on SO only
    SearchEnumMultiSelectField sensfTranType = new SearchEnumMultiSelectField();
    sensfTranType.operatorSpecified = true;
    sensfTranType.operator = SearchEnumMultiSelectFieldOperator.anyOf;
    String[] tranTypes = new String[1];
```
String tranType = "_salesOrder";
tranTypes[0] = tranType;
semsfTranType.searchValue = tranTypes;
tsb.type = semsfTranType;

// condition 2: tranId contains 182
SearchStringField sfTranId = new SearchStringField();
sfTranId.searchValue = "182"; // tranId contains 11
sfTranId.@operator = SearchStringFieldOperator.contains;
sfTranId.operatorSpecified = true;
tsb.tranId = sfTranId;

// condition 3: Is MultiShipping Routes enabled
SearchBooleanField sbfShipLineEnabled = new SearchBooleanField();
sbfShipLineEnabled.searchValue = true;
sbfShipLineEnabled.searchValueSpecified = true;
tsb.shipping = sbfShipLineEnabled;

// condition 4: Open SO
SearchBooleanField sbfTranStatus = new SearchBooleanField();
sbfTranStatus.searchValue = true;
sbfTranStatus.searchValueSpecified = true;
tsb.shipRecvStatusLine = sbfTranStatus;

// ts is the transaction search object

SOAP Request

<searchRecord xmlns:q1="urn:sales_2017_1.transactions.webservices.netsuite.com"
xsi:type="q1:TransactionSearchAdvanced">
  <q1:criteria>
    <q1:basic>
      <tranId operator="contains" xmlns="urn:common_2017_1.platform.webservices.netsuite.com">
        <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">182</searchValue>
      </tranId>
      <type operator="anyOf" xmlns="urn:common_2017_1.platform.webservices.netsuite.com">
        <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">_salesOrder</searchValue>
      </type>
    </q1:basic>
  </q1:criteria>
  <q1:columns>
    <q1:basic>
      <internalId xmlns="urn:common_2017_1.platform.webservices.netsuite.com">
        <customLabel xmlns="urn:core_2017_1.platform.webservices.netsuite.com">Sales Order ID</customLabel>
      </internalId>
      <shipRecvStatus xmlns="urn:common_2017_1.platform.webservices.netsuite.com">
        <customLabel xmlns="urn:core_2017_1.platform.webservices.netsuite.com">Order Fulfilled</customLabel>
      </shipRecvStatus>
      <shipRecvStatusLine xmlns="urn:common_2017_1.platform.webservices.netsuite.com">
      </shipRecvStatusLine>
    </q1:basic>
  </q1:columns>
</searchRecord>
SOAP Request (with Shipping and ShipRecvStatusLine criteria)

```xml
<searchRecord xmlns:q1="urn:sales_2017_1.transactions.webservices.netsuite.com" xsi:type="q1:TransactionSearchAdvanced">
    <q1:criteria>
        <q1:basic>
            <shipRecvStatus xmlns="urn:common_2017_1.platform.webservices.netsuite.com">
                <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">true</searchValue>
            </shipRecvStatus>
            <tranId operator="contains" xmlns="urn:common_2017_1.platform.webservices.netsuite.com">
                <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">11</searchValue>
            </tranId>
            <type operator="anyOf" xmlns="urn:common_2017_1.platform.webservices.netsuite.com">
                <searchValue xmlns="urn:core_2017_1.platform.webservices.netsuite.com">_salesOrder</searchValue>
            </type>
        </q1:basic>
    </q1:criteria>
    <q1:columns>
        <q1:basic>
            <internalId xmlns="urn:common_2017_1.platform.webservices.netsuite.com">
                <customLabel xmlns="urn:core_2017_1.platform.webservices.netsuite.com">Sales Order ID</customLabel>
            </internalId>
            <shipRecvStatus xmlns="urn:common_2017_1.platform.webservices.netsuite.com">
                <customLabel xmlns="urn:core_2017_1.platform.webservices.netsuite.com">Order Fulfilled</customLabel>
            </shipRecvStatus>
            <tranDate xmlns="urn:common_2017_1.platform.webservices.netsuite.com" />
        </q1:basic>
    </q1:columns>
</searchRecord>
```

SOAP Response

```xml
<platformCore:searchRow xmlns:platformCore="urn:common_2017_1.platform.webservices.netsuite.com" xsi:type="tranSales:TransactionSearchRow">
    <tranSales:basic xmlns:platformCommon="urn:common_2017_1.platform.webservices.netsuite.com">
        <platformCommon:internalId>
            <platformCore:searchValue internalId="986"/>
        </platformCommon:internalId>
        <platformCore:customLabel>Sales Order ID</platformCore:customLabel>
        <platformCommon:shipRecvStatusLine>
            <platformCore:searchValue>true</platformCore:searchValue>
        </platformCommon:shipRecvStatusLine>
        <platformCore:customLabel>Order Fulfilled</platformCore:customLabel>
    </tranSales:basic>
</platformCore:searchRow>
```
Programmatically manipulate data returned by a search

Java

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

    CustomerSearchAdvanced searchRecord = new CustomerSearchAdvanced();
    searchRecord.setSavedSearchId("26"); // A saved customer search

    SearchResult result = _port.search(searchRecord);

    if( result.getTotalRecords() > 0 ) {
        // retain the search ID to get more pages
        String sSearchId = result.getSearchId();

        SearchRowList rowList = result.getSearchRowList();

        processRowList(rowList);

        int iNumPages = result.getTotalPages();

        for ( int i=2; i<=iNumPages; i++) {
            result = _port.searchMoreWithId(sSearchId, i);
            rowList = result.getSearchRowList();

            processRowList(rowList);
        }
    }
}

public void processRowList(SearchRowList rowList) {
    for (int i=0; i<rowList.getSearchRow().length; i++) {
        CustomerSearchRow row = (CustomerSearchRow) rowList.getSearchRow(i);
        CustomerSearchRowBasic basic = row.getBasic();

        SearchColumnStringField companyName = basic.getCompanyName(0);
        _console.write("Company Name: "+companyName.getSearchValue());
        if (basic.getEmail(0).getSearchValue() != null) {
            String email = basic.getEmail(0).getSearchValue();
            _console.write("Email: "+email);
        }
        if (basic.getPhone(0).getSearchValue() != null) {
            String phone = basic.getPhone(0).getSearchValue();
            _console.write("Phone: "+phone);
        }
    }
}
```
Searching for a Multi-select Custom Field

In the following code sample, the results for the custom transaction field custcolcolumnname are returned.

**Java**

```java
// transaction search by custom column field
TransactionSearchBasic transactionSearch = new TransactionSearchBasic();

SearchCustomFieldList searchCustomFieldList = new SearchCustomFieldList();
transactionSearch.setCustomFieldList(searchCustomFieldList);

// make the multiselectsearch
SearchMultiSelectCustomField searchMultiSelectCustomField = new SearchMultiSelectCustomField();

ListOrRecordRef listOrRecordRef = new ListOrRecordRef();
listOrRecordRef.setInternalId("3"); // the internal id of the custom list value
listOrRecordRef.setType("1"); // your custom list typeId

searchCustomFieldList.setCustomField(new SearchCustomField[]{searchMultiSelectCustomField});

// make the search expression
searchMultiSelectCustomField.setscriptId("custcolcolumnname"); // the name of the tx custom column
searchMultiSelectCustomField.setOperator(SearchMultiSelectFieldOperator.anyOf);

SearchResult sr = _port.search(transactionSearch);
```

**C#**

```csharp
private void searchForMultiSelectCustomField()
{
    if (_isAuthenticated)
    {
        _out.info("\nExecuting search ..... \n");
        // transaction search by custom column field
        TransactionSearch transactionSearch = new TransactionSearch();
        TransactionSearchBasic transactionSearchBasic = new TransactionSearchBasic();

        //Java - the SearchCustomFieldList is not used.
        //SearchCustomFieldList searchCustomFieldList = new SearchCustomFieldList();
        //transactionSearch.setCustomFieldList(searchCustomFieldList);
        SearchMultiSelectCustomField searchMultiSelectCustomField = new
```
Search Multi Select Custom Field();

// make the search expression
// the name of the transaction custom column
searchMultiSelectCustomField.scriptId = "custbody_multi_select";
searchMultiSelectCustomField.@operator = SearchMultiSelectFieldOperator.anyOf;
searchMultiSelectCustomField.operatorSpecified = true;
// custom list called colors with typeId 1, values blue - internalId 1, green - id2 etc
// we are looking for transactions which have transaction body field
// of type multi select set to color blue
ListOrRecordRef listOrRecordRef = new ListOrRecordRef();
listOrRecordRef.internalId = "3";
listOrRecordRef.typeId = "1";

searchMultiSelectCustomField.searchValue = new ListOrRecordRef[] { listOrRecordRef };
SearchCustomField[] searchCustomFieldList = new SearchCustomField[] {
    searchMultiSelectCustomField
};

// Java
//searchCustomFieldList.setCustomField(new SearchCustomField[]{searchMultiSelectCustomField});
transactionSearchBasic.customFieldList = searchCustomFieldList;
transactionSearch.basic = transactionSearchBasic;
SearchResult searchRes = _service.search(transactionSearch);
_out.info("nSearch Result contains " + searchRes.totalRecords + " record(s) \n");
}

else
{
    out.info(
        \nCannot call search operation because there is no active session. " +
        "You must be first logged on before attempting to call saved search.\n"
    );
}

Search Issues and Best Practices for SOAP Web Services and SuiteScript

You may encounter issues when a SOAP web services or SuiteScript search executed in NetSuite returns multiple pages of results. These issues occur because records can be created, deleted, or updated as you page through the search results. You can avoid the issues described here by following the recommended best practices.

Potential Issues for SOAP Web Services and SuiteScript Searches

Consider the following possible issues when you execute a search in SOAP web services or SuiteScript that returns a large data set.

- If a record that originally fit the search criteria is updated and no longer fits the criteria, it is skipped by the search. In this case, some records that were originally included in the search results may not be returned, and the total number of returned records may be smaller than the original total number of results.
If new records are created that fit the search criteria, they are also returned. The total number of returned records may be higher after you page through the results than the original total number of results, and duplicate results may be returned.

Results may be missing. For example, if a record from the first results page is updated and it no longer fits the criteria, another record now fits into page one. However, because earlier pages are skipped when going through subsequent pages of search results, this record is not returned.

In SOAP web services, a greater or smaller number of records may be returned after you page through the search results than the number of records that originally fit the search criteria.

Page two and the subsequent pages of the results set may contain fewer records than the specified page size.

### Best Practices for Searches in SOAP Web Services and SuiteScript

To avoid the issues associated with paging through search results, and for more reliable search results, consider the following best practices.

- Avoid using loosely defined search criteria that return a large data set. If a large data set is returned and you must go through the results page by page, records may be missing or duplicated, because the data may keep changing as you go through the pages of search results.
- Make your search criteria as specific as possible. With more specific criteria, the result set is smaller and there is less chance that records fitting the search criteria are created or updated as you go through the pages of the result set.
- If your search results fit into one page, you can avoid the issues that may occur when you page through the result set.
- Similarly, if you use more specific search criteria, there is less chance that records are updated in a way that they no longer fit the criteria. You can avoid the issue of records missing from the result set.
- In SuiteScript 2.0, as an alternative to using the N/search module, you can use the searchTask object members and searchTaskStatus object members of the N/task Module when you search for large data sets. Note that searchTask is an asynchronous search.
- By creating a hybrid integration that uses SOAP web services and RESTlets, you can also use the N/task module as an alternative to SOAP web services search.

### searchMore

The searchMore operation is used to retrieve more records after an initial search operation.

**Important:** Users who authenticate to NetSuite through login can use either searchMore or searchMoreWithId to paginate through search results. Users who authenticate to NetSuite by providing user credentials in the header of their SOAP requests can use only searchMoreWithId to paginate through search results, since searchMore requires an active session.

**Note:** (See Request-Level Credentials for information on request-level-credential authentication.)

The results returned in a searchMore operation reflect the records in the next segment as defined in the original search operation. Therefore, if a record is deleted before all records have been returned, the total number of records may differ from the total record count returned in the search operation.
— the deleted record is not returned. If a record is **added**, that record is not returned in subsequent searchMore operations. However, the data in each record returned is current such that if a change to a record occurs in between the time of the original search operation and the searchMore operation, the updated data is returned.

**Example**

Suppose you submit the following request for all employees whose firstName contains 'Susan' and set the pagesize to 10 records:

```xml
<soap:Header>
  <platformMsgs:searchPreferences>
    <platformMsgs:bodyFieldsOnly>true</platformMsgs:bodyFieldsOnly>
    <platformMsgs:pageSize>10</platformMsgs:pageSize>
  </platformMsgs:searchPreferences>
</soap:Header>
<soap:Body>
  <platformMsgs:search>
    <platformMsgs:searchRecord xsi:type="listEmp:EmployeeSearch">
      <listEmp:firstName operator="contains" xsi:type="platformCore:SearchStringField">
        <platformCore:searchValue>Susan</platformCore:searchValue>
      </listEmp:firstName>
    </platformMsgs:searchRecord>
  </platformMsgs:search>
</soap:Body>
```

As illustrated in the following figure, if the database contains 12 matches at the time of the initial Search, the Search result would provide the first 10 records as defined in the pageSize preference. A subsequent SearchMore operation would return only the remaining two records from the original search result even if another employee record, Susan Cook, is added prior to the SearchMore request being submitted.

**Request**

The SearchMoreRequest type is used for the request. It contains the following fields.
<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageIndex</td>
<td>SearchRecord</td>
<td>An index that specifies which page in the search to return. If it is not provided, the next page is returned.</td>
</tr>
</tbody>
</table>

**Response**

The SearchMoreResponse type is used for the response. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this search. All applicable errors or warnings will be listed within this type.</td>
</tr>
<tr>
<td>totalRecords</td>
<td>xsd:int</td>
<td>The total number of records for this search. Depending on the pageSize value, some or all the records may be returned in this response</td>
</tr>
<tr>
<td>pageSize</td>
<td>xsd:int</td>
<td>The page size for this search.</td>
</tr>
<tr>
<td>totalPages</td>
<td>xsd:int</td>
<td>The total number of pages that are part of this search.</td>
</tr>
<tr>
<td>pageIndex</td>
<td>xsd:int</td>
<td>The page index for the current set of results.</td>
</tr>
<tr>
<td>recordList</td>
<td>Record[]</td>
<td>A list of records that meet the criteria for this search. The actual records returned need to be of a type that extends the abstract type of record.</td>
</tr>
</tbody>
</table>

**Faults**

This operation can throw one of the following faults. See [SOAP Fault Status Codes](#) for more information on faults.

- InvalidCredentialsFault
- InvalidSessionFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

**Sample Code**

**SOAP Request**

In this example, page 2 of the search result set is requested.

```xml
<soap:Body>
<platformMsgs:searchMore>
  <platformMsgs:pageIndex>2</platformMsgs:pageIndex>
</platformMsgs:searchMore>
```
<soapenv:Body>
<searchMoreResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<searchResult xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
<status isSuccess="true"/>
<totalRecords>93</totalRecords>
<pageSize>10</pageSize>
<totalPages>10</totalPages>
<pageIndex>2</pageIndex>
<recordList>
<record internalId="80" xsi:type="ns1:Customer"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns1="urn:relationships_2017_1.lists.webservices.netsuite.com">
<ns1:entityId>Jackson Alexander</ns1:entityId>
<ns1:isInactive>false</ns1:isInactive>
...[more fields]
</record>
<ns1:customFieldList>
<customField internalId="custentity_map" xsi:type="StringCustomFieldRef">
<value>http://maps.google.com</value>
</customField>
</ns1:customFieldList>
</record>
<record internalId="227" xsi:type="ns2:Customer"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns2="urn:relationships_2017_1.lists.webservices.netsuite.com">
<ns2:entityId>Seena Thomas</ns2:entityId>
<ns2:isInactive>false</ns2:isInactive>
<ns2:companyName>Seena Thom Inc.</ns2:companyName>
...[more fields]
</record>
<ns2:customFieldList>
<customField internalId="custentity_map" xsi:type="StringCustomFieldRef">
<value>http://maps.google.com</value>
</customField>
</ns2:customFieldList>
</record>
...[more records]
</recordList>
</searchResult>
</searchMoreResponse>
</soapenv:Body>

C#

private void searchMore( SearchResult response )
```java
public void searchMore(SearchResult result) throws RemoteException,
        ExceededUsageLimitFault, UnexpectedErrorFault, InvalidSessionFault,
        ExceededRecordCountFault {
    // SearchResult response;
    boolean isGetAllPages = false;

    // Keep getting pages until there are no more pages to get
    while (result.getPageSize() != null
            && result.getPageIndex() != null
            && (result.getTotalRecords().intValue() > (result.getPageSize().intValue() * result.getPageIndex().intValue()))) {
        if (!isGetAllPages) {
            // Only continue if user wants to get the next page
            _out.write( "\nThere are more search results. Would you like to get the next page for this search? (A/Y/N): " );
            String userResponse = _out.readLn().ToUpper();
            if (String.Equals(userResponse, "N")) {
                break;
            } else if (String.Equals(userResponse, "A")) {
                isGetAllPages = true;
            }
        }

        // Invoke searchMore() operation
        response = _service.searchMore( response.pageIndex + 1 );

        // Process response
        if (response.status.isSuccess) {
            processCustomerSearchResponse( response );
        } else {
            _out.error( getStatusDetails( response.status ) );
        }
    }
}
```
There are more search results. Would you like to get the next page for this search? (A/Y/N): 

String userResponse = _console.readLn().toUpperCase();
if ("N".equals(userResponse)) {
    break;
} else if ("A".equals(userResponse)) {
    isGetAllPages = true;
}

// Invoke searchMore() operation
result = _port.searchMore(result.getPageIndex().intValue() + 1);

// Process result
if (result.getStatus().isIsSuccess()) {
    processCustomerSearchResponse(result);
} else {
    _console.error(getStatusDetails(result.getStatus()));
}

// searchMoreWithId

searchMoreWithId

Users who authenticate to NetSuite by providing their credentials in the SOAP header of their requests must use searchMoreWithId to retrieve search results that span multiple pages. They cannot use searchMore or searchNext, since both operations require an active session. (For information on request-level-credential authentication, see Request-Level Credentials.)

Note: Users who authenticate to NetSuite through login can use searchMore or searchMoreWithId to paginate through search results.

The searchMoreWithId operation allows you to reference a specific search result set by its searchId, a parameter included in all search results. As with searchMore, you must set the pageIndex value to specify which page in the search to return.

Usage Notes

Consider the following information when you use the searchMoreWithId operation.

- When you paginate through the search results, the searchMoreWithId operation triggers a new search for the page specified by the pageIndex value. If records included in the search result set are updated while you go through the result pages, and they do not match the earlier search criteria anymore, they will be omitted from the results displayed on that page you specified. To avoid this, you need to ensure that records are not modified while you are calling a search operation.
- Search IDs expire if they have not been used within 15 minutes after their creation. Passing an expired or invalid searchId will return search results with a “failed” status and StatusDetailCode=INVALID_JOB_ID.
- There is no async equivalent for this operation.
For information about achieving the best search performance, see Search Issues and Best Practices for SOAP Web Services and SuiteScript.

Request

The SearchMoreWithIdRequest type is used for the request. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>searchId</td>
<td>string</td>
<td>The search result ID.</td>
</tr>
<tr>
<td>pageIndex</td>
<td>int</td>
<td>An index that specifies which page in the search to return.</td>
</tr>
</tbody>
</table>

Response

The SearchMoreWithIdResponse type is used for the response. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this search. All applicable errors or warnings will be listed within this type.</td>
</tr>
<tr>
<td>totalRecords</td>
<td>xsd:int</td>
<td>The total number of records for this search. Depending on the pageSize value, some or all the records may be returned in this response</td>
</tr>
<tr>
<td>pageSize</td>
<td>xsd:int</td>
<td>The page size for this search.</td>
</tr>
<tr>
<td>totalPages</td>
<td>xsd:int</td>
<td>The total number of pages that are part of this search.</td>
</tr>
<tr>
<td>pageIndex</td>
<td>xsd:int</td>
<td>The page index for the current set of results.</td>
</tr>
<tr>
<td>searchId</td>
<td>string</td>
<td>Returns a specific search based on its search ID.</td>
</tr>
<tr>
<td>recordList</td>
<td>Record[]</td>
<td>A list of records that meet the criteria for this search. The actual records returned need to be of a type that extends the abstract type of record.</td>
</tr>
<tr>
<td>searchRowList</td>
<td>SearchRowList</td>
<td>A list of return columns that meet the criteria for this search.</td>
</tr>
</tbody>
</table>

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
Sample Code

The following samples show the SOAP for the initial search as well as the SOAP for subsequent searchMoreWithId operation used to specify the next page of the search. The ID returned in the initial search is specified as the searchId when executing searchMoreWithId.

Note: Prefix-to-namespace mappings have been omitted for readability.

SOAP Request for Initial Search

```xml
<soapenv:Envelope>
  <soapenv:Header>
    <platformMsgs:searchPreferences>
      <platformMsgs:bodyFieldsOnly>true</platformMsgs:bodyFieldsOnly>
      <platformMsgs:pageSize>100</platformMsgs:pageSize>
    </platformMsgs:searchPreferences>
    <platformMsgs:passport>
      <platformCore:email>jdoe@netsuite.com</platformCore:email>
      <platformCore:password>mypassword</platformCore:password>
      <platformCore:account>000034</platformCore:account>
      <platformCore:role internalId="37"/>
    </platformMsgs:passport>
  </soapenv:Header>
  <soapenv:Body>
    <platformMsgs:search>
      <searchRecord xsi:type="platformCommon:ContactSearchBasic">
        <platformCommon:city operator="is" xsi:type="platformCore:SearchStringField">
          <platformCore:searchValue xsi:type="xsd:string">San Francisco</platformCore:searchValue>
        </platformCommon:city>
      </searchRecord>
    </platformMsgs:search>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response for Initial Search

```xml
<soapenv:Envelope>
  <soapenv:Header>
    <platformMsgs:documentInfo>
      <platformMsgs:nsId>WEBSERVICES_528736_07012008543995006307049233_d53ef4d2273b15</platformMsgs:nsId>
    </platformMsgs:documentInfo>
  </soapenv:Header>
  <soapenv:Body>
    <platformMsgs:searchResponse>
      <platformCore:searchResult>
        <!-- Search results here -->
      </platformCore:searchResult>
    </platformMsgs:searchResponse>
  </soapenv:Body>
</soapenv:Envelope>
```
SOAP Request for Getting the Next Page Using searchMoreWithId

```xml
<soapenv:Envelope>
  <soapenv:Header>
    <platformMsgs:searchPreferences>
      <platformMsgs:bodyFieldsOnly>true</platformMsgs:bodyFieldsOnly>
      <platformMsgs:pageSize>100</platformMsgs:pageSize>
    </platformMsgs:searchPreferences>
    <platformMsgs:passport>
      <platformCore:email>jdoe@netsuite.com</platformCore:email>
      <platformCore:password>mypassword</platformCore:password>
      <platformCore:account>000034</platformCore:account>
      <platformCore:role internalId="37"/>
    </platformMsgs:passport>
  </soapenv:Header>
  <soapenv:Body>
    <platformMsgs:searchMoreWithId>
      <searchId>WEBSERVICES_528736_07012008543995006307049233_d53ef4d2273b15</searchId>
      <pageIndex>2</pageIndex>
    </platformMsgs:searchMoreWithId>
  </soapenv:Body>
</soapenv:Envelope>
```

SOAP Response for Getting the Next Page Using searchMoreWithId

```xml
<soapenv:Envelope>
  <soapenv:Header>
    <platformMsgs:documentInfo>
      <platformCore:status isSuccess="true"/>
      <platformCore:totalRecords>231</platformCore:totalRecords>
      <platformCore:pageSize>100</platformCore:pageSize>
      <platformCore:totalPages>3</platformCore:totalPages>
      <platformCore:pageIndex>1</platformCore:pageIndex>
      <platformCore: searchId > WEBSERVICES_528736_07012008543995006307049233_d53ef4d2273b15
      <platformCore: searchId >
      <platformCore:recordList>
        <platformCore:record internalId="4" externalId="entity-4" xsi:type="listRel:Contact">
          <listRel:entityId>john</listRel:entityId>
          <listRel:firstName>John</listRel:firstName>
          ....
        </platformCore:record>
      </platformCore:recordList>
    </platformMsgs:documentInfo>
  </soapenv:Header>
</soapenv:Envelope>
```
Java

```
public List<Contact> searchContactsByCity(NetSuitePortType port, String city) throws Exception {
    SearchResult searchResult = null;         // search result for each page
    Record [] contacts = null;                    // contacts for each page
    List<Contact> consolidatedResults = new ArrayList<Contact>();   // to return

    // build search criteria
    ContactSearchBasic contactSearch = new ContactSearchBasic();
    contactSearch.setCity(new SearchStringField(city, SearchStringFieldOperator.is));

    // execute the search
    searchResult = port.search(contactSearch);

    if( searchResult.getTotalRecords() > 0 )
    {
        // retain the search ID  to get more pages
        String sSearchId = searchResult.getSearchId();

        // process first page
        contacts = searchResult.getRecordList().getRecord();
        for (Record contact : contacts)
            consolidatedResults.add((Contact)contact);

        // process remaining pages using search ID
        int iNumPages = searchResult.getTotalPages();
        for ( int i=2; i<=iNumPages; i++)
        {
            //
        }
    }
    return consolidatedResults;
}
```
// get the page
searchResult = port.searchMoreWithId(sSearchId, i);

// process the page
contacts = searchResult.getRecordList().getRecord();
for (Record contact : contacts)
    consolidatedResults.add((Contact)contact);
}
}
return consolidatedResults;

searchNext

The searchNext operation is used to retrieve the next set of records after an initial search operation.

⚠️ **Important:** Users who authenticate to NetSuite through login can use either searchNext or searchMoreWithId to paginate through search results. Users who authenticate to NetSuite by providing user credentials in the header of their SOAP requests can use only searchMoreWithId to paginate through search results, since searchNext requires an active session.

💡 **Note:** (See Request-Level Credentials for information on request-level-credential authentication.)

Request

The SearchNextRequest type is used for the request. It does not contain any fields.

Response

The SearchNextResponse type is used for the response. It contains the following fields:

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>status</td>
<td>Status</td>
<td>The status for this search. All applicable errors or warnings will be listed within this type.</td>
</tr>
<tr>
<td>totalRecords</td>
<td>xsd:int</td>
<td>The total number of records for this search. Depending on the pageSize value, some or all the records may be returned in this response.</td>
</tr>
<tr>
<td>pageSize</td>
<td>xsd:int</td>
<td>The page size for this search.</td>
</tr>
<tr>
<td>totalPages</td>
<td>xsd:int</td>
<td>The total number of pages that are part of this search.</td>
</tr>
<tr>
<td>pageIndex</td>
<td>xsd:int</td>
<td>The page index for the current set of results.</td>
</tr>
<tr>
<td>searchId</td>
<td>string</td>
<td>Returns a specific search based on its search ID.</td>
</tr>
<tr>
<td>recordList</td>
<td>Record[]</td>
<td>A list of records that meet the criteria for this search. The actual records returned need to be of a type that extends the abstract type of record.</td>
</tr>
<tr>
<td>searchRowList</td>
<td>SearchRowList</td>
<td>A list of return columns that meet the criteria for this search.</td>
</tr>
</tbody>
</table>
Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidCredentialsFault
- InvalidSessionFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request

```xml
<soap:Body>
   <platformMsgs:searchNext/>
</soap:Body>
```

SOAP Response

```xml
<soapenv:Body>
<searchNextResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<searchResult xmlns="urn:core_2017_1.platform.webservices.netsuite.com">
<status isSuccess="true"/>
<totalRecords>93</totalRecords>
<pageSize>10</pageSize>
<totalPages>10</totalPages>
<pageIndex>3</pageIndex>
<recordList>
  <record internalId="865" xsi:type="ns1:Customer"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:ns1="urn:relationships_2017_1.lists.webservices.netsuite.com">
    <ns1:entityId>John Bauer</ns1:entityId>
    <ns1:isInactive>false</ns1:isInactive>
    ...[more fields]
  </record>
  ...[more records]
</recordList>
</searchResult>
</searchNextResponse>
```

SuiteTalk SOAP Web Services Platform Guide
ssoLogin

The ssoLogin operation provides a mechanism for a partner application to log in to NetSuite on behalf of the user. This approach enables login to take place without the user’s credentials ever going through the partner servers. To use this operation, the inbound single sign-on feature must be enabled.

When submitting SOAP web services requests, the NetSuite server can accept any SOAP document as long as it is valid against the NetSuite WSDL, and it does not matter how the SOAP document was generated. You must ensure that all cookies received from the server are managed and passed back to the server in subsequent calls. There are a total of three cookies that are initially returned in the HTTP header of the login response, including the JSESSIONID. Make sure that you do not use the JSESSIONID cookie when invoking the ssoLogin operation.

For inbound single sign-on to work, a single sign-on mapping must exist for every user that accesses NetSuite through the external application. This mapping can be created manually when the user logs in, or by using the mapSso operation.

**Important:** As of 2018.2, new Inbound Single Sign-on (SSO) solutions are no longer enabled for use with SOAP web services for new customers or new partners. Current partners may continue to sell their existing Inbound SSO solutions to new customers. However, it is strongly recommended that you consider other options, and use token-based authentication (TBA) if possible. For more information, see Token-Based Authentication Details.

The following steps outline the general workflow that occurs the first time the user logs in, if the user is creating the mapping manually.

1. User authenticates at the partner site.
2. Partner provides the single sign-on link to user.
3. Partner produces a token. The token is generated from a string that includes three pieces of data: a timestamp, the remote company ID from the external system, and the remote user ID from the external system. This string is encrypted by using a private key.
4. The token enables the user to be redirected to NetSuite.
5. The user logs in to NetSuite, and the authentication link for the user is created.
6. After the single sign-on mapping is created, the partner invokes ssoLogin operation via SOAP web services to access NetSuite on behalf of the user.

For more details about the inbound SSO process, see the help topic Inbound Single Sign-on.

Please note the following important details about the ssoLogin operation:

- SsoPassports cannot be handled at the request level.
- If you want to use inbound single sign-on, you must purchase the “Single Sign On Configuration” line item. Contact your account manager for assistance.
- The SOAP mapSso operation is not directly connected to the ssoLogin operation. The purpose of mapSso is to establish the mapping between a user’s identity in an external application and the user’s
The ssoLogin operation is for establishing **inbound** connections only. To create an outbound connection that goes from NetSuite to a third-party application, see the help topic [Outbound Single Sign-on (SuiteSignOn)](https://www.netsuite.com/DeveloperCenter/soap/help/topics/outbound-sso.html).

**Important:** NetSuite hosts customer accounts in multiple data centers. For that reason, the correct URL for SOAP web services access varies depending on the data center hosting the account.

For details about dynamic discovery of URLs, see [Effects of Accounts' Host Data Center on URLs](https://www.netsuite.com/DeveloperCenter/soap/help/topics/accounts-url-discovery.html) and [Dynamic Discovery of URLs for SOAP Web Services and RESTlet Clients](https://www.netsuite.com/DeveloperCenter/soap/help/topics/dynamic-url-discovery.html).

### Request

The `SsoPassportRequest` type is used for this request. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SsoPassport</td>
<td><code>SsoPassport</code></td>
</tr>
</tbody>
</table>

The following table describes the fields in the `SsoPassport` complex type. All are required.

<table>
<thead>
<tr>
<th>Field</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>authenticationToken</code></td>
<td>The <code>authenticationToken</code> is generated by using three values: a timestamp, the <code>partnerAccount</code> value, and the <code>partnerUserId</code> value. For more information about generating a token, see <a href="https://www.netsuite.com/DeveloperCenter/soap/help/topics/sample-token-generation.html">Example Single Sign-on Token and Redirect URLs</a>.</td>
</tr>
<tr>
<td><code>partnerId</code></td>
<td>This value identifies the partner, or external system. NetSuite uses this data to retrieve the decryption key needed to process the token.</td>
</tr>
<tr>
<td><code>partnerAccount</code></td>
<td>This value is sometimes referred to as the remote company ID. It is used to identify the user in the external system.</td>
</tr>
<tr>
<td><code>partnerUserId</code></td>
<td>This value is sometimes referred to as the remote user ID. It is used to identify the user in the external system.</td>
</tr>
</tbody>
</table>

*Note:* Starting with the 2015.2 endpoint, this parameter is required.

### Response

The `SsoLoginResponse` type is used for the response.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>status</code></td>
<td><code>Status</code></td>
<td>The status for this operation. All applicable errors or warnings are listed within this type.</td>
</tr>
</tbody>
</table>

SuiteTalk SOAP Web Services Platform Guide
<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>wsRoleList</td>
<td>WsRoleList</td>
<td>Returns a list of roles available for the specified user.</td>
</tr>
</tbody>
</table>

**Faults**

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidAccountFault
- InvalidCredentialsFault
- InvalidVersionFault
- ExceededRequestLimitFault
- UnexpectedErrorFault

**Sample Code**

The following snippets show how to use the ssoLogin operation.

**SOAP Request**

```
<urn:ssoLogin xmlns:urn="urn:messages_2017_1.platform.webservices.netsuite.com"
               xmlns:urn1="urn:core_2017_1.platform.webservices.netsuite.com">
  <urn:ssoPassport>
    <urn1:authenticationToken>57E1A7DA1CD637E6D0F6D7AF3F9EE9680DE6D1E0D3376786086E4448760CEC5D249031CA0B4618EB50C7317030D3E27158F663C7AC11D384</urn1:authenticationToken>
    <urn1:partnerId>1</urn1:partnerId>
    <urn1:partnerAccount>35185</urn1:partnerAccount>
    <urn1:partnerUserId>john.smith</urn1:partnerUserId>
  </urn:ssoPassport>
</urn:ssoLogin>
```

**Java**

```java
public void ssoLogin() throws Exception {
    String myPartnerAccount = "35185";
    String myPartnerUserId = "john.smith";

    SsoPassport sso = new SsoPassport();
    sso.setPartnerId("1");
    sso.setPartnerAccount(myPartnerAccount);
    sso.setPartnerUserId(myPartnerUserId);

    sso.setAuthenticationToken(generateToken(myPartnerAccount, myPartnerUserId));
}
```
update

The update operation is used to update an instance of a record in NetSuite. It is similar to the updateList operation, which allows users to update more than one record at a time.

Only the fields that have been populated in each submitted record are updated in the system. If a field has NOT been populated, it is not updated in the system and it retains its previous value. If a field is set to an empty string, the previous value of the field is replaced with an empty string. Therefore, when updating records, it is recommended that you get the desired record, instantiate a new record of the same type, populate only the fields that require an update and then submit the updated record. This ensures that only the fields requiring an update are written on submission.

**Important:** Calculated and hidden fields in records are always updated by the system unless your service explicitly overrides the system values. For more information, see Hidden Fields. Also, custom fields can only be set to NULL by submitting the field in nullFieldList. For more information, see CustomFieldList.

To ensure that the most recent data for a specific record is being modified, when a Web service request is received, the values for that record are retrieved at the time of the Update request rather than with the initial Get of the associated record. The record is then updated by the values submitted in the request. It is possible that between the time of the retrieval of the record field values and the submission of the updated fields that the record is altered from another source (for example from a UI submission). In this case an error message is returned to indicate that the fields have been modified since your service retrieved the record.

Although records of a particular type may be used in multiple integration scenarios, each record instance can only have a single external ID value. To maintain data integrity, only a single integrated application can set and update external ID values for each record type. External ID values for all records of a particular type must all come from the same external application.

**Note:** As of the 2011.2 endpoint, when a record is updated and the values that are sent are exactly the same as the existing record values, the record is not reset (nothing happens). In previous endpoints, sometimes records were reset in these cases.

Request

The UpdateRequest type is used for the request. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>Record</td>
<td>Contains an array of record objects. The record type is an abstract type so an instance of a type that extends record must be used—such as Customer or Event.</td>
</tr>
</tbody>
</table>

Response

The UpdateResponse type is used for the response. It contains the following fields.
<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>WriteResponse</td>
<td>Contains details on the status of the operation and a reference to the updated record.</td>
</tr>
</tbody>
</table>

## Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

## Sample Code

### SOAP Request

In the following example, a customer's companyName is updated. The internal ID for the customer must be provided in the request.

```xml
<soap:Body>
<platformMsgs:update>
  <platformMsgs:record internalId="980" xsi:type="listRel:Customer">
    <listRel:companyName>Shutter Fly Corporation</listRel:companyName>
  </platformMsgs:record>
</platformMsgs:update>
</soap:Body>
```

### SOAP Response

```xml
<updateResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <writeResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"/>
    <baseRef internalId="980" type="customer" xsi:type="ns2:RecordRef"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
  </writeResponse>
</updateResponse>
```
private void updateCustomer()
{
    // This operation requires a valid session
    this.login( true );

    // Get nsKey for update
    _out.write( "Enter nsKey for customer record to be updated : " );
    customer.internalId = _out.readLine().ToUpper();

    // Set name and email
    customer.entityId = "XYZ 2 Inc";
    customer.companyName = "XYZ 2, Inc.";
    customer.email = "bsanders@xyz.com";

    // Populate the address. Updating a list through WS results in the
    // entire contents of the previous list being replaced by the new
    // list.
    CustomerAddressbook address = new CustomerAddressbook();
    address.defaultBilling = true;
    address.defaultBillingSpecified = true;
    address.defaultBilling = false;
    address.defaultBillingSpecified = true;
    address.label = "Billing Address";
    address.addr1 = "4765 Sunset Blvd";
    address.city = "San Mateo";
    address.state = "CA";
    address.country = Country._unitedStates;

    // Attach the address to the customer
    CustomerAddressbookList addressList = new CustomerAddressbookList();
    CustomerAddressbook[] addresses = new CustomerAddressbook[1];
    addresses[0] = address;
    addressList.addressbook = addresses;
    customer.addressbookList = addressList;

    // Invoke add() operation
    WriteResponse response = _service.update( customer );

    // Process the response
    if ( response.status.isSuccess )
    {
        _out.info("The following customer was updated successfully:
                  key=" + ((RecordRef) response.baseRef).internalId + 
                  "entityId=" + customer.entityId + 
                  "companyName=" + customer.companyName + 
                  "email=" + customer.email + 
                  "addressbookList[0].label=" + customer.addressbookList.addressbook[0].label );
    }
    else
    {

}
public void updateCustomer() throws RemoteException,
ExceededUsageLimitFault, UnexpectedErrorFault, InvalidSessionFault,
ExceededRecordCountFault {
    // This operation requires a valid session
    this.login(true);

    Customer customer = new Customer();

    // Get nsKey for update
    _console.write("Enter nsKey for customer record to be updated : ");
    customer.setInternalId(_console.readLn().toUpperCase());

    // Set name and email
    customer.setEntityId("XYZ 2 Inc");
    customer.setCompanyName("XYZ 2, Inc.");
    customer.setEmail("bsanders@xyz.com");

    // Populate the address. Updating a list through WS results in the
    // entire contents of the previous list being replaced by the new
    // list.
    CustomerAddressbook address = new CustomerAddressbook();
    address.setDefaultBilling(Boolean.TRUE);
    address.setDefaultShipping(Boolean.FALSE);
    address.setLabel("Billing Address");
    address.setAdd1("4765 Sunset Blvd");
    address.setCity("San Mateo");
    address.setState("CA");
    address.setCountry(Country._unitedStates);

    // Attach the address to the customer
    CustomerAddressbookList addressList = new CustomerAddressbookList();
    CustomerAddressbook[] addresses = new CustomerAddressbook[1];
    addresses[0] = address;
    addressList.setAddressbook(addresses);
    customer.setAddressbookList(addressList);

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

    // Process the response
    if (response.getStatus().isIsSuccess()) {
        _console.info("The following customer was updated successfully:
          
          \nkey=" + ((RecordRef) response.getBaseRef()).getInternalId()
          + "\nentityId=" + customer.getEntityId()
          + "\ncompanyName=" + customer.getCompanyName());
+ "\email="
+ customer.getEmail()
+ "\addressbookList[0].label="
+ customer.getAddressbookList().getAddressbook(0)
+ .getLabel());
} else {
  _console.error(getStatusDetails(response.getStatus()));
}
}

Updating Record Lists

When updating a list of records (a sublist) within a business record you CANNOT update a specific item in the list. Instead you must interact with the sublist as a whole. For details, see Sublists in SOAP Web Services.

updateList

The updateList operation is used to update one or more instances of a record type in NetSuite.

If there are multiple records, they can either be of the same record type or different record types. For example, it's possible to update a customer and a contact within a single request using this operation.

Only the fields that have been populated in each submitted record are updated in the system. If a field has not been populated, it is not updated in the system and it retains its previous value. If a field is set to an empty string, the previous value of the field is replaced with an empty string.

Although records of a particular type may be used in multiple integration scenarios, each record instance can only have a single external ID value. To maintain data integrity, only a single integrated application can set and update external ID values for each record type. External ID values for all records of a particular type must all come from the same external application.

⚠️ Important: Calculated and hidden fields in records are always updated by the system unless your service explicitly overrides the system values. For more information, refer to Hidden Fields.

ℹ️ Note: As of the 2011.2 endpoint, when a record is updated and the values that are sent are exactly the same as the existing record values, the record is not reset (nothing happens). In previous endpoints, sometimes records were reset in these cases.

Request

The UpdateListRequest type is used for the request. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>record[]</td>
<td>Record</td>
<td>Contains an array of record objects. The record type is an abstract type so an instance of a type that extends record must be used—such as Customer or Event.</td>
</tr>
</tbody>
</table>
Note: An asynchronous equivalent is available for this operation, asyncUpdateList. For information about asynchronous request processing, see Synchronous Versus Asynchronous Request Processing.

Response

The UpdateListResponse type is used for the response. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>response[]</td>
<td>WriteResponse</td>
<td>Contains an array of WriteResponse objects, each of which contains details on the status of that update operation and a reference to the created record.</td>
</tr>
</tbody>
</table>

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In the following example, two customer records are updated. The first has the single field companyName updated, whereas the second updates two fields: entityId and companyName. The internalID for each record must be provided and the type of record (Customer) to be updated.

```xml
<soap:Body>
<platformMsgs:updateList>
  <platformMsgs:record internalId="980" xsi:type="listRel:Customer">
    <listRel:companyName>Shutter Fly Corporation</listRel:companyName>
  </platformMsgs:record>
  <platformMsgs:record internalId="981" xsi:type="listRel:Customer">
    <listRel:entityId>GNCC</listRel:entityId>
    <listRel:companyName>GNCC Corp</listRel:companyName>
  </platformMsgs:record>
</platformMsgs:updateList>
</soap:Body>
```
SOAP Response

```xml
<soapenv:Body>
<updateListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<writeResponseList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
<writeResponse>
<ns1:status isSuccess="true" xmlns:ns1="urn:core_2017_1.platform.webservices.netsuite.com"/>
;baseRef internalId="980" type="customer" xsi:type="ns2:RecordRef"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns2="urn:core_2017_1.platform.webservices.netsuite.com"/>
</writeResponse>
<writeResponse>
<ns3:status isSuccess="true" xmlns:ns3="urn:core_2017_1.platform.webservices.netsuite.com"/>
;baseRef internalId="981" type="customer" xsi:type="ns4:RecordRef"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns4="urn:core_2017_1.platform.webservices.netsuite.com"/>
</writeResponse>
</writeResponseList>
</updateListResponse>
</soapenv:Body>
```

C#

```csharp
private void updateCustomerList()
{
    // This operation requires a valid session
    this.login( true );

    // Prompt for list of nsKeys and put in an array
    _out.write( "Enter nsKeys for customer records to be updated (separated by commas): " );
    String reqKeys = _out.readLn();
    string [] nsKeys = reqKeys.Split( new Char[] {','} );

    // Create an array of Record objects to hold the customers
    Record[] records = new Record[nsKeys.Length];

    // For each submitted nsKey, populate a customer object
    for ( int i=0; i<nsKeys.Length; i++ )
    {
        Customer customer = new Customer();

        // Update name
        customer.entityId = "XYZ Inc " + i;
        customer.companyName = "XYZ, Inc. " + i;
        customer.internalId = nsKeys[i].Trim();
        records[i] = customer;
    }

    // Invoke updateList() operation to update customers
    WriteResponse[] responses = _service.updateList( records );

    // Process responses for all successful updates
    _out.info( "The following customers were updated successfully:" );
}
updateList

```java
bool hasFailures = false;
for ( int i=0; i<responses.Length; i++ ) {
    if ( (responses[i] != null) && (responses[i].status.isSuccess) ) {
        _out.info( "\nCustomer[" + i + "]:" );
        _out.info( "key=" + ((RecordRef) responses[i].baseRef).internalId +
"entityId=" + ((Customer) records[i]).entityId +
"companyName=" + ((Customer) records[i]).companyName );
    } else {
        hasFailures = true;
    }
}

// Process responses for all unsuccessful updates
if ( hasFailures ) {
    _out.info( "\nThe following customers were not updated:\n" );
    for ( int i=0; i<responses.Length; i++ ) {
        if ( (responses[i] != null) && (!responses[i].status.isSuccess) ) {
            _out.info( "Customer[" + i + "]:" );
            _out.info( "key=" + ((RecordRef) responses[i].baseRef).internalId );
            _out.errorForRecord( getStatusDetails( responses[i].status ) );
        }
    }
}
}
```

Java

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

    // Prompt for list of nsKeys and put in an array
    _console
    .write("Enter nsKeys for customer records to be updated (separated by commas): ");
    String reqKeys = _console.readLn();
    String[] nsKeys = reqKeys.split(",");

    // Create an array of Record objects to hold the customers
    Record[] records = new Record[nsKeys.length];

    // For each submitted nsKey, populate a customer object
    for (int i = 0; i < nsKeys.length; i++) {
        Customer customer = new Customer();
```
updateList

```java
// Update name
customer.setEntityId("XYZ Inc " + i);
customer.setCompanyName("XYZ, Inc. " + i);

customer.setInternalId(nsKeys[i].trim());
records[i] = customer;
}

// Invoke updateList() operation to update customers
WriteResponseList responseList = _port.updateList(records);

// Process responses for all successful updates
WriteResponse[] responses = responseList.getWriteResponse();
boolean hasFailures = false;
_console.info("\nThe following customers were updated successfully:");
for (int i = 0; i < responses.length; i++) {
    if ((responses[i] != null) && (responses[i].getStatus().isIsSuccess())) {
        _console.info("Customer[" + i + "]:");
        _console.info("key=");
        + ((RecordRef) responses[i].getBaseRef()).getInternalId()
        + ";entityId="
        + ((Customer) records[i]).getEntityId()
        + ";companyName="
        + ((Customer) records[i]).getCompanyName());
    } else {
        hasFailures = true;
    }
}

// Process responses for all unsuccessful updates
if (hasFailures) {
    _console.info("\nThe following customers were not updated:");
    for (int i = 0; i < responses.length; i++) {
        if ((responses[i] != null) && (!responses[i].getStatus().isIsSuccess())) {
            _console.info("Customer[" + i + "]:");
            _console.info("key=");
            + ((RecordRef) responses[i].getBaseRef()).getInternalId();
            _console.errorForRecord(getStatusDetails(responses[i].getStatus()));
        }
    }
}
```

updateInviteeStatus

The updateInviteeStatus operation lets users respond to NetSuite events that have been sent to them. This operation takes both the internal ID of the event as well as a calendar event status as arguments.

After invitees have responded to the event invitation, the Event record is updated with their response. Possible responses include accepted, declined, tentative, and noResponse.

Note the following:
- For information on the Event record in SOAP web services, see the help topic Events (CalendarEvent). For general information on scheduling events in NetSuite, see the help topic Scheduling Events.
- There is no async equivalent for the updateInviteeStatus operation.

Note the following when using this operation:

- Users must have a valid session. This operation includes a passport header to support request level credentials. (For information on request level authentication, see Request-Level Credentials.)
- To update other properties on the Event record, the event owner should still use the update operation. Note, however, event owners are not exempt from using updateInviteeStatus if they have to update their own event response status. There may be cases in which event owners must decline their own event and have another person run the event for them.
- Unlike in the NetSuite UI (see figure), invitees will not be able to send a message back to the organizer. SOAP web services does not currently support messages attached to events. An event invitee's response can include only one of the following values: _accepted, _declined, _tentative, _noResponse.

### Request

The UpdateInviteeStatusRequest type is used for the request.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventId</td>
<td>RecordRef</td>
<td>References an existing instance of an Event record.</td>
</tr>
<tr>
<td>responseCode</td>
<td>CalendarEventAttendeeResponse</td>
<td>The CalendarEventAttendeeResponse type includes the following enums:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>_accepted</td>
</tr>
<tr>
<td></td>
<td></td>
<td>_declined</td>
</tr>
<tr>
<td></td>
<td></td>
<td>_tentative</td>
</tr>
<tr>
<td></td>
<td></td>
<td>_noResponse</td>
</tr>
</tbody>
</table>

### Response

The UpdateInviteeStatusResponse type is used for the response.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>WriteResponse</td>
<td>Contains details on the status of the operation and a reference to the updated record.</td>
</tr>
</tbody>
</table>
Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

C#

```csharp
NetSuiteService nss = new NetSuiteService();   // login details omitted
UpdateInviteeStatusReference inviteeStatusRef = new UpdateInviteeStatusReference();

// Set the event id for the status update
RecordRef eventIdRef = new RecordRef();
eventIdRef.internalId = "100";   // Substitute this with your own event id
inviteeStatusRef.eventId = eventIdRef;

// Set the event status
inviteeStatusRef.responseCode = CalendarEventAttendeeResponse._accepted;

// Update invitee event status
WriteResponse resp = nss.updateInviteeStatus(inviteeStatusRef);
```

SOAP Request

```xml
<updateInviteeStatus xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <updateInviteeStatusReference>
        <eventId internalId="100" xmlns="urn:core_2017_1.platform.webservices.netsuite.com" />
        <responseCode xmlns="urn:core_2017_1.platform.webservices.netsuite.com">_accepted</responseCode>
    </updateInviteeStatusReference>
</updateInviteeStatus>
```

SOAP Response

```xml
<updateInviteeStatusResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <writeResponse>
        <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
    </writeResponse>
</updateInviteeStatusResponse>
```
updateInviteeStatusList

The updateInviteeStatusList operation is used to update one or more NetSuite events. For general details on the updateInviteeStatus operation, see updateInviteeStatus.

Request

The UpdateInviteeStatusListRequest type is used for the request.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>updateInviteeStatusReference[]</td>
<td>UpdateInviteeStatusReference</td>
<td>UpdateInviteeStatusReference uniquely identifies the event and the status you want to update the event with.</td>
</tr>
</tbody>
</table>

The UpdateInviteeStatusReference type includes the following elements:

- eventId
- responseCode

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventId</td>
<td>RecordRef</td>
<td>References an existing instance of an Event record.</td>
</tr>
<tr>
<td>responseCode</td>
<td>CalendarEventAttendeeResponse</td>
<td>The CalendarEventAttendeeResponse type includes the following enums:</td>
</tr>
<tr>
<td></td>
<td>_accepted</td>
<td></td>
</tr>
<tr>
<td></td>
<td>_declined</td>
<td></td>
</tr>
<tr>
<td></td>
<td>_tentative</td>
<td></td>
</tr>
<tr>
<td></td>
<td>_noResponse</td>
<td></td>
</tr>
</tbody>
</table>

Response

The UpdateInviteeStatusListResponse type is used for the response.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>response[]</td>
<td>WriteResponse</td>
<td>Contains an array of WriteResponse objects, each of which contains details on the status of that updateInviteeStatusList operation and a reference to each event record.</td>
</tr>
</tbody>
</table>

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

Sample Code

C#

```csharp
UpdateInviteeStatusReference[] inviteeStatusRefList = new UpdateInviteeStatusReference[2];
for (int i=0; i<2; i++)
{
    RecordRef eventIdRef = new RecordRef();
    if (i==0)
        eventIdRef.internalId = "100";
    else
        eventIdRef.internalId = "101";
    UpdateInviteeStatusReference statusRef = new UpdateInviteeStatusReference();
    statusRef.eventId = eventIdRef;
    statusRef.responseCode = CalendarEventAttendeeResponse._accepted;
    inviteeStatusRefList[i] = statusRef;
}
nss.updateInviteeStatusList(inviteeStatusRefList);
```

SOAP Request

```xml
<updateInviteeStatusList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <updateInviteeStatusReference>
        <eventId internalId="100" xmlns="urn:core_2017_1.platform.webservices.netsuite.com" />
        <responseCode xmlns="urn:core_2017_1.platform.webservices.netsuite.com">_accepted</responseCode>
    </updateInviteeStatusReference>
    <updateInviteeStatusReference>
        <eventId internalId="101" xmlns="urn:core_2017_1.platform.webservices.netsuite.com" />
        <responseCode xmlns="urn:core_2017_1.platform.webservices.netsuite.com">_accepted</responseCode>
    </updateInviteeStatusReference>
</updateInviteeStatusList>
```

SOAP Response

```xml
<updateInviteeStatusListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <writeResponseList>
        <writeResponse>
            <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            <baseRef internalId="100" type="calendarEvent" xsi:type="platformCore:RecordRef"/>
        </writeResponse>
    </writeResponseList>
</updateInviteeStatusListResponse>
```
The upsert operation is used to add a new instance or to update an instance of a record in NetSuite. It is similar to the upsertList operation, which allows users to add or update more than one record at a time.

The upsert operation is similar to both the add and update operations, but upsert can be run without first determining whether a record exists in NetSuite. A record is identified by its external ID and its record type. If a record of the specified type with a matching external ID exists in the system, it is updated. If it does not exist, a new record is created.

Because external ID is mandatory for this operation, upsert is supported only for records that support the external ID field. Also, this operation prohibits the passing of internal ID values.

**Note:** To prevent duplicate records, NetSuite recommends using external IDs and the upsert and upsertList operations to add records to NetSuite.

### Limitations on upsert

- Although records of a particular type may be used in multiple integration scenarios, each record instance can only have a single external ID value. To maintain data integrity, only a single integrated application can set and update external ID values for each record type. External ID values for all records of a particular type must all come from the same external application.
- Upsert cannot complete updates during an `initialize / initializeList` operation, when a record is transformed into a record of another type. In this case, upsert only adds and does not update records.
- Updates through the upsert operation are also subject to the same limitations as updates through the update operation. For details of these limitations, see update.

### Request

The UpsertRequest type is used for the request. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>record</td>
<td>Record</td>
<td>Contains an array of record objects. The record type is an abstract type so an instance of a type that extends record must be used—such as Customer or Event.</td>
</tr>
</tbody>
</table>

### Response

The UpsertResponse type is used for the response. It contains the following fields.
<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>WriteResponse</td>
<td>Contains details on the status of the operation and a reference to the created or updated record.</td>
</tr>
</tbody>
</table>

**Faults**

- This operation can throw one of the following faults. See [SOAP Fault Status Codes](#) for more information.
  - InvalidSessionFault
  - InvalidCredentialsFault
  - ExceededRequestLimitFault
  - ExceededUsageLimitFault
  - ExceededRecordCountFault
  - ExceededRequestSizeFault
  - UnexpectedErrorFault

- This operation returns the following run-time error if the passed record type does not support the external ID field:
  ```
  INVALID_RCRD_TYPE:  <record_type> does not support external ID and cannot be used with upsert
  ```

- This operation returns the following run-time error if passed data includes internal ID:
  ```
  USER_ERROR:  You cannot set internalId with upsert.
  ```

- This operation returns the following run-time error if passed data does not include external ID:
  ```
  USER_ERROR:  This operation requires a value for externalId.
  ```

For more information about error codes, see [Error Status Codes](#).

**Sample Code**

**SOAP Request**

```xml
<soap:Body>
  <upsert xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
    <record xmlns:q1="urn:relationships_2017_1.lists.webservices.netsuite.com" xsi:type="q1:Customer" externalId="THISISMYEXTID">
      <q1:entityId>XYZ 2 Inc</q1:entityId>
      <q1:companyName>XYZ 2, Inc.</q1:companyName>
      <q1:email>bsanders@xyz.com</q1:email>
    </record>
  </upsert>
</soap:Body>
```

**SOAP Response**

```xml
<soapenv:Body>
</soapenv:Body>
```
C#  
private void upsertCustomer()  
{  
    // This operation requires a valid session  
    this.login(true);  
    
    Customer customer = new Customer();  
    
    // Get externalId for upsert  
    _out.write("Enter externalId for customer record to be created or updated : ");  
    customer.externalId = _out.readLine().ToUpper();  
    
    // Set name and email  
    customer.entityId = "XYZ 2 Inc";  
    customer.companyName = "XYZ 2, Inc.";  
    customer.email = "bsanders@xyz.com";  
    
    // Invoke upsert() operation  
    WriteResponse response = _service.upsert(customer);  
    
    // Process the response  
    if (response.status.isSuccess )  
    {  
        _out.info(  
            "The upsert operation was successful :" +  
            "internalId=" + ((RecordRef) response.baseRef).internalId +  
            "externalId=" + ((RecordRef) response.baseRef).externalId +  
            "entityId=" + customer.entityId +  
            "companyName=" + customer.companyName);  
    } else {  
        _out.error( getStatusDetails( response.status ) );  
    }  
}  

Java  
public void upsertCustomer() throws RemoteException,ExceededUsageLimitFault,  
UnexpectedErrorFault, InvalidSessionFault,ExceededRecordCountFault  
{  
    // This operation requires a valid session  
    this.login(true);  
    Customer customer = new Customer();
import _console, _port, customer = Record.create(Record.PARTNER马来); // Get externalId for add or update _console.write("\nEnter externalId for customer record to be updated : "); customer.setExternalId(_console.readLn());

// Set name and email
customer.setEntityId("XYZ 2 Inc");
customer.setCompanyName("XYZ 2, Inc.");
customer.setEmail("bsanders@xyz.com");

// Invoke upsert() operation
WriteResponse response = _port.upsert(customer);

// Process the response
if (response.getStatus().isIsSuccess())
{
    _console.info("\nThe following customer was created/updated successfully:"
    + "\nkey=" + ((RecordRef) response.getBaseRef()).getInternalId()
    + "\nexternalId=" + ((RecordRef) response.getBaseRef()).getExternalId()
    + "\nentityId=" + customer.getEntityId()
    + "\ncompanyName=" + customer.getCompanyName()
    + "\nemail=" + customer.getEmail());
}
else
{
    _console.error(getStatusDetails(response.getStatus()));
}

--

The upsertList operation is used to add or update one or more instances of a record type in NetSuite.

If there are multiple records, they can either be of the same record type or different record types. For example, it's possible to add or update a customer and a contact within a single request using this operation.

The upsertList operation is similar to both the addList and updateList operations, but upsert can be run without first determining whether records exist in NetSuite. Records are identified by their external ID and their record type. If a record of the specified type with a matching external ID exists in the system, it is updated. If it does not exist, a new record is created.

Because external ID is mandatory for this operation, upsertList is supported only for records that support the external ID field. Also, this operation prohibits the passing of internal ID values.

Note: To prevent duplicate records, NetSuite recommends using external IDs and the upsert and upsertList operations to add records to NetSuite.

Limitations on upsertList

- Although records of a particular type may be used in multiple integration scenarios, each record instance can only have a single external ID value. To maintain data integrity, only a single integrated
application can set and update external ID values for each record type. External ID values for all records of a particular type must all come from the same external application.

- UpsertList cannot complete updates during an initialize / initializeList operation, when a record is transformed into a record of another type. In this case, upsertList only adds and does not update records.
- Updates through the upsertList operation are also subject to the same limitations as updates through the updateList operation. For details of these limitations, see updateList.

Request

The UpsertListRequest type is used for the request. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>record[]</td>
<td>Record</td>
<td>Contains an array of record objects. The record type is an abstract type so an instance of a type that extends record must be used—such as Customer or Event.</td>
</tr>
</tbody>
</table>

Note: An asynchronous equivalent is available for this operation, asyncUpsertList. For information about asynchronous request processing, see Synchronous Versus Asynchronous Request Processing.

Response

The UpsertListResponse type is used for the response. It contains the following fields.

<table>
<thead>
<tr>
<th>Element Name</th>
<th>XSD Type</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>response[]</td>
<td>WriteResponse</td>
<td>Contains an array of WriteResponse objects, each of which contains details on the status of that upsert operation and a reference to the created or updated record.</td>
</tr>
</tbody>
</table>

Faults

This operation can throw one of the following faults. See SOAP Fault Status Codes for more information on faults.

- InvalidSessionFault
- InvalidCredentialsFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestSizeFault
- UnexpectedErrorFault

This operation returns the following run-time error if the passed record type does not support the external ID field:
INVALID_RCRD_TYPE: <record_type> does not support external ID and cannot be used with upsert

This operation returns the following run-time error if passed data includes internal ID:

USER_ERROR: You cannot set internalId with upsert.

This operation returns the following run-time error if passed data does not include external ID:

USER_ERROR: This operation requires a value for externalId.

For more information about error codes, see Error Status Codes.

Sample Code

**SOAP Request**

```xml
<soap:Body>
    <upsertList xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
        <record xmlns:q1="urn:relationships_2017_1.lists.webservices.netsuite.com" xsi:type="q1:Customer" externalId="ext1">
            <q1:entityId>XYZ Inc 0</q1:entityId>
            <q1:companyName>XYZ, Inc. 0</q1:companyName>
        </record>
        <record xmlns:q2="urn:relationships_2017_1.lists.webservices.netsuite.com" xsi:type="q2:Customer" externalId="ext2">
            <q2:entityId>XYZ Inc 1</q2:entityId>
            <q2:companyName>XYZ, Inc. 1</q2:companyName>
        </record>
    </upsertList>
</soap:Body>
```

**SOAP Response**

```xml
<soapenv:Body>
    <upsertListResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
        <writeResponseList>
            <writeResponse>
                <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
                <baseRef internalId="970" externalId="ext1" type="customer" xsi:type="platformCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </writeResponse>
            <writeResponse>
                <platformCore:status isSuccess="true" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
                <baseRef internalId="974" externalId="ext2" type="customer" xsi:type="platformCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
            </writeResponse>
        </writeResponseList>
    </upsertListResponse>
</soapenv:Body>
```
private void upsertCustomerList()
{
    // This operation requires a valid session
    this.login( true );

    // Prompt for list of externalIds and put in an array
    _out.write( "Enter externalIds for customer records to be updated (separated by commas):" );
    String reqKeys = _out.readLn();
    string[] nsKeys = reqKeys.Split( new Char[] {','} );

    // Create an array of Record objects to hold the customers
    Record[] records = new Record[nsKeys.Length];

    // For each submitted nsKey, populate a customer object
    for ( int i=0; i<nsKeys.Length; i++ )
    {
        Customer customer = new Customer();

        // Update name
        customer.entityId = "XYZ Inc " + i;
        customer.companyName = "XYZ, Inc. " + i;
        customer.externalId = nsKeys[i].Trim();
        records[i] = customer;
    }

    // Invoke upsertList() operation to create or update customers
    WriteResponse[] responses = _service.upsertList( records );

    // Process responses for all successful updates
    _out.info( "The following customers were updated or created successfully:" );

    bool hasFailures = false;
    for ( int i=0; i<responses.Length; i++ )
    {
        if ( (responses[i] != null) && (responses[i].status.isSuccess) )
        {
            _out.info( "Customer[" + i + "]:" );
            _out.info( "internalId=" + ((RecordRef) responses[i].baseRef).internalId + 
                       "externalId=" + ((RecordRef) responses[i].baseRef).externalId + " 
                       "entityId=" + ((Customer) records[i]).entityId + 
                       "companyName=" + ((Customer) records[i]).companyName );
        }
        else
        {
            hasFailures = true;
        }
    }

    // Process responses for all unsuccessful updates
}
if ( hasFailures )
{
    _out.info( "\nThe following customers were not updated:\n" );
    for ( int i=0; i<responses.Length; i++ )
    {
        if ( (responses[i] != null) && (!responses[i].status.isSuccess) )
        {
            _out.info( "Customer[" + i + "]:" );
            _out.info( "key=" + ((RecordRef) responses[i].baseRef).internalId );
            _out.errorForRecord( getStatusDetails(responses[i].status ) );
        }
    }
}
}

Java

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

    // Prompt for list of externalIds and put in an array
    _console.write("\nEnter externalIds for customer records to be updated (separated by commas): ");
    String reqKeys = _console.readLn();
    String[] nsKeys = reqKeys.split(",");

    // Create an array of Record objects to hold the customers
    Record[] records = new Record[nsKeys.length];

    // For each submitted nsKey, populate a customer object
    for (int i = 0; i < nsKeys.length; i++)
    {
        Customer customer = new Customer();

        // Update name
        customer.setEntityId("XYZ Inc " + i);
        customer.setCompanyName("XYZ, Inc. " + i);
        customer.setExternalId(nsKeys[i].trim());
        records[i] = customer;
    }

    // Invoke upsertList() operation to create or update customers
    WriteResponseList responseList = _port.upsertList(records);

    // Process responses for all successful upserts
    WriteResponse[] responses = responseList.getWriteResponse();
    boolean hasFailures = false;
    _console.info("\nThe following customers were processed successfully:");

for (int i = 0; i < responses.length; i++)
{
    if ((responses[i] != null) & (responses[i].getStatus().isIsSuccess()))
    {
        _console.info("\nCustomer[" + i + "]:");
        _console.info("key=" + ((RecordRef)
responses[i].getBaseRef()).getInternalId() + "]externalId=" + ((RecordRef)
responses[i].getBaseRef()).getExternalId() + "]entityId=" + ((Customer)
records[i]).getEntityId() + "]companyName=" + ((Customer)
records[i]).getCompanyName());
    } else
    {
        hasFailures = true;
    }

    // Process responses for all unsuccessful updates
    if (hasFailures)
    {
        _console.info("The following customers were not updated:\n");
        for (int i = 0; i < responses.length; i++)
        {
            if ((responses[i] != null) & (!
responses[i].getStatus().isIsSuccess()))
            {
                _console.info("Customer[" + i + "]:");
                _console.info("key=" + ((RecordRef)
responses[i].getBaseRef()).getInternalId());
                _console.errorForRecord(getStatusDetails(responses[i].getStatus()));
            }
        }
    }
}
SOAP Web Services PHP Toolkit

If you are new to PHP and SOAP web services, you should review the following topics:

- PHP Toolkit Overview
- Downloading the PHP Toolkit
- Configuring an Environment for the PHP Toolkit
- Creating a SOAP Web Services PHP Project
- Creating and Submitting Records Using the PHP Toolkit
- Logging SOAP Requests and Responses Using the PHP Toolkit

If you run into issues during your development, see Troubleshooting PHP and SOAP Web Services.

⚠️ **Important:** Using PHP scripts with SOAP web services requires PHP version 5.6 or later.

PHP Toolkit Overview

NetSuite provides a library and tools to assist in the development of PHP applications that interface to the SOAP web services platform.

The NetSuite PHP toolkit for SOAP web services is a core library containing all classes, objects and methods parsed from the WSDL of a SOAP web services endpoint. The toolkit also comes with several helper functions that can be used to simplify working with arrays and objects.

⚠️ **Important:** Each PHP Toolkit is specific to a SOAP web services endpoint. You must download the correct version of the toolkit for the endpoint you are using. Also, updating to a new endpoint requires an update of the toolkit. To use the 2019.2 PHP Toolkit, you must upgrade to the 2019.2 WSDL. The 2019.2 PHP Toolkit does not officially support different WSDL versions.

Updating Existing PHP Code

NetSuite offered a completely revamped PHP Toolkit as part of 2012.2. As of 2012.2, the PHP Toolkit has the ability to generate proxy classes for every new NetSuite WSDL, creating a development environment similar to Axis or .NET.

Additionally, the new toolkit uses a different programming model (the use of proxy classes over pure associative arrays, though the latter is still supported).

If you upgrade to any version of the PHP Toolkit later than 2012.2, existing PHP applications created using previous versions of the toolkit will need to be updated. To understand the nature of the coding changes, see the table below. Also see the sample applications that come with the new toolkit.

<table>
<thead>
<tr>
<th>Record object initialization</th>
<th>PHP Toolkit 2012.2 and Later</th>
<th>PHP Toolkit Versions Older than 2012.2</th>
</tr>
</thead>
</table>
| Each record type can now be initialized using the same record type name. | Initialize the object using the object nsComplexObject. | Example - To initialize a Customer record object: 
$CustomerRec = new Customer(); |
| Example - To initialize a Customer record object: 
$CustomerRec = new nsComplexObject('Customer'); | Example - To initialize a Customer record object: 
$CustomerRec = new nsComplexObject('Customer'); |
<p>| Object type Initialization | Each complex, search and custom field type is initialized using the same object type name: | These objects are initialized using the object nsComplexObject. |</p>
<table>
<thead>
<tr>
<th>PHP Toolkit 2012.2 and Later</th>
<th>PHP Toolkit Versions Older than 2012.2</th>
</tr>
</thead>
</table>
| All object types are listed and explained in Platform Enumerations. | Example - To initialize a basic search for customers:  
$selectTypeField = new RecordRef();  
To initialize a recordRef object:  
$selectTypeField = new RecordRef(); |

| Calling the NetSuite SOAP web services operations | The operations are called from the service object instantiated from the NetSuiteService class (NetSuiteService.php).  
Example -  
$service = new NetSuiteService();  
...  
$addResponse = $service->add($request); |

| The setFields method | This is a method of the record object used to set the array of field values to the record.  
Example -  
$customerFields = array (  
'isPerson' => true,  
'firstName' => $firstName,  
'lastName' => $lastName,  
'companyName' => $companyName  
);  
setFields($customerRec, $customerFields); |

| Request object | You must instantiate a request object depending on the kind of operation. This request object will accept the Record object, which will then be set to the operation. Each operation has a corresponding request type.  
▪ AddRequest for add operation  
▪ AddListRequest for addList operation  
▪ SearchRequest for search operation.  
Example - To add a customer:  
$service = new NetSuiteService();  
$CustomerRec = new Customer();  
...  
$request = new AddRequest();  
$request->record = $CustomerRec;  
...  
$addResponse = $service->add($request); |

| | None. The record object is directly set to the operation.  
Example - To add a customer:  
$CustomerRec = new nsComplexObject('Customer');  
...  
$addResponse = $myNSclient->add($CustomerRec); |
PHP Toolkit Overview

343

343

Downloading the PHP Toolkit

To download the 2019.2 PHP Toolkit, click the link for the PHPToolkit_2019_2.zip file.

The toolkit is provided as an archive file and contains:

- A folder called PHPToolkit that contains:
  - NetSuiteService.php – Class library
  - NSconfig.php – Passport configuration file
  - NSPHPClient.php – Helper functions
- A folder called Samples that contains PHP code examples that are usable in your own projects.

Next see, Configuring an Environment for the PHP Toolkit.

Configuring an Environment for the PHP Toolkit

You can run PHP scripts from the command line by installing a PHP interpreter. Command line scripts are suitable for stand-alone applications and are useful for scripting and testing. See Installing PHP for the Command Line for more details.

The PHP Toolkit's real power, however, is in the ability to use a web server-based PHP interpreter module to integrate NetSuite into an existing web presence, whether it is an external e-commerce website, or an intranet-only Human Resources portal, for example.

A web server executes a PHP script through the included PHP interpreter module. PHP scripts can be text files containing PHP code, or they can be HTML web pages containing PHP tags indicating sections of PHP code. See Installing a Web Server for PHP for more details.

Note: For information on downloading the PHP Toolkit, see Downloading the PHP Toolkit.

Installing PHP for the Command Line

Many distributions of Linux and Mac OS X® are pre-installed with PHP. Packages for Windows® can be obtained from http://windows.php.net/download/.

After PHP is installed, you need to edit the php.ini file. The PHP Toolkit needs to use certain libraries that are not enabled in a PHP installation by default.

In the php.ini configuration file, in the section [Dynamic Extensions], the following lines need to be added if they do not exist:

```
extension=php_soap.dll
extension=php_openssl.dll
```

The modules indicated by the .dll extensions pertains to a Windows installation. If you are configuring a Unix-based platform, change .dll to .so:
Configuring an Environment for the PHP Toolkit

```
extension=php_soap.so
extension=php_openssl.so
```

**Note:** For a Windows installation of PHP, change the caching directory to something similar to:

```
soap.wsdl_cache_dir="c:/Windows/Temp"
```

## Installing a Web Server for PHP

**Note:** This section talks about using Apache Web Server, but any web server that supports PHP, SSL, and SOAP should be able to be used for SOAP web services.

The simplest way to provision a web server for the PHP Toolkit is to install a LAMP package. LAMP is the acronym for Linux, Apache, MySQL, and PHP. After a LAMP package is installed, a complete web server with database and PHP scripting functionality is available with little configuration required. The PHP Toolkit does require, however, that you make a few modifications to the default LAMP installation.

**Note:** There are several LAMP distributions available free of charge for most computer architectures. A well known package for Windows® is WAMP, which is available at [http://www.wampserver.com/en/](http://www.wampserver.com/en/). Follow the instructions that come with the package to install and initially configure it.

After LAMP is installed, the php.ini and the httpd.conf configuration files need to be edited. Note that the LAMP installation will not use any existing PHP installation; it will install its own self-contained PHP package.

In the php.ini configuration file, in the section [Dynamic Extensions], the following lines need to be added if they do not exist:

```
extension=php_soap.dll
extension=php_openssl.dll
```

The modules indicated by the .dll extensions pertains to a Windows installation. If you are configuring a Unix-based platform, change .dll to .so:

```
extension=php_soap.so
extension=php_openssl.so
```

The Apache Web Server needs to be told to load the PHP module so that it can run PHP scripts. In the httpd.conf configuration file in the section Dynamic Shared Object Support, the following line must be added. (Note that the line may already exist; in that case check that the path to the PHP module is correct.)

```
LoadModule php5_module C:\<PATH TO PHP>\modules\libphp5.dll
```

The module indicated by the .dll extension pertains to a windows installation. If you are configuring a Unix-based platform, the PHP module will have a .so extension. Note also that the path to the PHP module will differ:

```
LoadModule php5_module /<PATH TO PHP>/bin/php/modules/libphp5.so
```
Configuring an Environment for the PHP Toolkit

**Important:** As indicated in these examples, the name of the module is libphp5.dll or libphp5.so, but be aware that SOAP web services requires PHP 5.6 or later. In other words, the name of the module is not specific to the exact PHP version. (However, the exact version is likely indicated by the <PATH TO PHP> folder name.)

**Important:** After a change to any configuration files, LAMP's services must be restarted.

**Important:** If you are running PHP scripts from the command line and through a web server, there will be two php.ini files that need to be edited, one for the command line PHP installation, and one provided by the LAMP installation.

**Note:** For a Windows installation of PHP, change the caching directory to something similar to:

```
soap.wsdl_cache_dir="c:/Windows/Temp"
```

Creating a SOAP Web Services PHP Project

For each project you need to perform the following steps:

1. Copy the PHPToolkit directory to the directory in your IDE where your project resides.
2. Edit the NSconfig.php file, substituting your own account details.
3. If SOAP logging is required, create a folder called `nslog` inside the PHPToolkit folder. See Logging SOAP Requests and Responses Using the PHP Toolkit for more details.
4. Each solution needs to include the NetSuiteService.php file by adding the following line to the main source file:

   ```
   require_once '../PHPToolkit/NetSuiteService.php';
   ```

   This step will prompt most IDEs to parse the NetSuiteService.php file for auto completion information.
5. Instantiate a service:

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

   **Important:** Each time you upgrade your NetSuite endpoint, for example from WSDL version 2019.1 to 2019.2, you will also need upgrade the NetSuiteService.php class library to take advantage of any new features.

In 2016.1, Token-based Authentication support was added to the PHPToolkit. There is a sample `token_based_authentication.php` that details how to use token-based authentication with PHPToolkit in the PHPToolkit_2019_2.zip Samples folder. The following guidelines should be followed when using token-based authentication with PHPToolkit:

- Each request requires a unique TokenPassport
- NetSuite does not generate TokenPassports. You must provide your own implementation of `iTokenPassportGenerator`
- You configure NetSuite to use your `iTokenPassportGenerator` by calling the `setTokenGenerator`:

SuiteTalk SOAP Web Services Platform Guide
For more information about token-based authentication, see the help topic Token-based Authentication and Web Services.

Creating and Submitting Records Using the PHP Toolkit

These sections provide basic examples for creating and submitting NetSuite records using the PHP toolkit.

- Creating a Record with PHP
- Creating a Record with the setFields Method
- Submitting a Record

Creating a Record with PHP

The following example shows how to create and submit a Purchase Order. In this case, the Purchase Order object is created then populated one field at a time.

Note that after the Purchase Order is instantiated, the IDE should automatically start suggesting members of the PurchaseOrder object using its code completion mechanism.

```php
// Include the PHP Toolkit Library
require_once '../PHPToolkit/NetSuiteService.php';

// Instantiate a service
$service = new NetSuiteService();

// Create Purchase Order object
$po = new PurchaseOrder();

// Create entity record
$po->entity = new RecordRef();

// PO internal ID = 38
$po->entity->internalId = 38;

// Create PO Item sublist
$po->itemList = new PurchaseOrderItemList();

// Create PO Item
$poi = new PurchaseOrderItem();

// Set Item details
$poi->item->internalId = 15;
$poi->quantity = 5;
```
Creating a Record with the `setFields` Method

An alternative method to create the object is to use array notation to set the object fields. The advantage is more compact code; the disadvantage is that code completion will not work.

The following example creates the same Purchase Order object using the `setFields` method:

```php
// Create the Purchase Order object
$po = new PurchaseOrder();

// Create array of fields
$purchaseOrderFields = array('entity' => array('internalId' => 38), 'itemList' => array('item' => array( array('item' => array('internalId' => 15), 'quantity' => 5) )));

// Call setFields to recursively create objects and populate their fields
setFields($po, $purchaseOrderFields);
```

In addition to creating and populating objects, the `setFields` method performs basic type and field checking to make sure the array structure corresponds to the expected object hierarchy and types.

**Note:** The `setFields` method is defined in the `NSPHPClient.php` helper file, which is included automatically by `NetSuiteService.php`.

Submitting a Record

After the object is created and populated, the object can be submitted to NetSuite in one of the operations supported by the type of object you have created. The following code sample shows the `add` operation:

```php
// Create Add request
$request = new AddRequest();

// Attach the Purchase Order to the request
$request->record = $po;

// Submit the record and save the response
$addResponse = $service->add($request);
```

The `add` operation returns a `ReadResponse` object that is used to access the result of the operation. The following code sample reads and displays a simple textual indication of the result:

```php
if (!$addResponse->writeResponse->status->isSuccess)
{
    echo "ADD ERROR";
}
else
{
    echo "ADD SUCCESS, id " . $addResponse->writeResponse->baseRef->internalId;
}
```
Note: The PHP Toolkit package contains more code examples in the Samples folder.

Logging SOAP Requests and Responses Using the PHP Toolkit

To enable SOAP logging with the PHP Toolkit, create a folder called nslog in the PHPToolkit folder inside the project. The PHP Toolkit will automatically spool the SOAP requests generated by the PHP code, and the responses that have been returned by the server, into the nslog folder in standard XML format.

The following listing shows the typical project structure when logging is enabled:

```
Project_1 <Directory>
    PHPToolkit.php
    NSconfig.php
    NetSuiteService.php
    nslog <Directory>
        20120317.160138.6941-add-request.xml
        20120317.165167.8345-add-response.xml
        20120317.180653.3467-delete-request.xml
        20120317.187474.3356-delete-response.xml
        ...
```

Note: To disable logging, rename or delete the nslog folder.

Troubleshooting PHP and SOAP Web Services

The following table provides solutions for problems that may be encountered when using PHP with NetSuite’s SOAP web services.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>The following error is returned when attempting to send requests over https:</td>
<td>Edit the php.ini file by uncommenting “extension=php_openssl.dll” under Dynamic Extensions. Windows only: Make sure that libeay32.dll and ssleay32.dll files are in your path. You can do this by copying the files to your System32 folder.</td>
</tr>
<tr>
<td>The following timeout error is returned:</td>
<td>Edit the php.ini file and setting default_socket_timeout = 200.</td>
</tr>
<tr>
<td>Fatal error: Maximum execution time of 30 seconds exceeded.</td>
<td></td>
</tr>
<tr>
<td>The following error message is returned:</td>
<td></td>
</tr>
<tr>
<td>The following error message is returned:</td>
<td></td>
</tr>
<tr>
<td>Warning: It is not yet possible to assign complex types to properties.</td>
<td>Open the php.ini file and edit error_reporting. Set it to E_ERROR instead of E_ALL.</td>
</tr>
<tr>
<td>The execution of script stops and error was not shown, or notices are being shown on the page.</td>
<td>Open the php.ini file and set error_reporting = E_ERROR &amp; ~E_NOTICE.</td>
</tr>
<tr>
<td>Problem</td>
<td>Solution</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
</tr>
<tr>
<td>The following error is returned:</td>
<td>Open the php.ini file and set memory_limit = 80M.</td>
</tr>
<tr>
<td>Fatal error: Allowed memory size of 8388608 bytes exhausted (tried to allocate 3024 bytes).</td>
<td></td>
</tr>
<tr>
<td>Performance issue:</td>
<td>For a non-Windows installation of PHP, look for property soap.wsdl_cache_dir in the php.ini file and make sure that the specified folder (“/tmp” -&gt; c:\tmp) exists in the host.</td>
</tr>
<tr>
<td>Every time a SOAP web services request executes, PHP performs a GET on the wsdl.</td>
<td></td>
</tr>
<tr>
<td>You have made a change in your php.ini file, yet the value you set is not respected.</td>
<td>Make sure that the property exists only one time in the php.ini file.</td>
</tr>
</tbody>
</table>

**Note:** For a Windows installation of PHP, change the caching directory to something similar to:

```
soap.wsdl_cache_dir="c:/Windows/Temp"
```
SOAP Web Services Error Handling and Error Codes

This section provides information on the following:

- The three exception types that are supported in SOAP web services: warnings, errors, and faults. See SOAP Web Services Warnings, Errors, and Faults for an overview.
- SOAP Faults that can be thrown for each operation. See SOAP Faults for Each Operation.
  - SOAP Fault Status Codes
  - Error Status Codes
  - Warning Status Codes

SOAP Web Services Warnings, Errors, and Faults

Based on the error code or fault that is received, the client can take the appropriate action.

- **Warnings**: informational notifications requiring an action within the UI but requiring no response from a Web service call. Data may or may not be processed depending on preference settings. For more information on preferences, see Company-Wide Preferences.
- **Errors**: exceptions returned on a record-by-record basis due to invalid or incomplete data. The service is processed as requested, however, only those records without errors are updated.
- **Faults**: a fundamental exception type that results in the entire request not being processed.

**Warnings**

A warning is a notification sent to a user to prevent a subsequent error or to ensure better data quality. In the Netsuite UI, a warning is presented to the user through a dialog box and generally requires an action from the user. Since there is no interaction of this nature in the SOAP web services model, the request must specify what to do in the case of a warning. The options are:

- Ignore the warning and submit the record to the database
- Heed the warning and abort the submission — treat as an error

You can set a company wide preference on the SOAP Web Services Preferences page on how to handle warnings or you can specify how warnings should be handled in a specific request. Request level preferences override company-wide preferences. Refer to Company-Wide Preferences for information on how to set company-wide preferences.

Following is an example of a warning message in the response to a request to add a customer record without a zip code (with the Treat Warnings as Errors preference set to false):

**Code**

```xml
<addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
  <writeResponse>
    <platformCore:status isSuccess="true" xmlns:platformCore="urn:core.platform_2017.1.platform.webservices.netsuite.com">
      <platformCore:statusDetail type="WARN">
        <platformCore:code>WARNING</platformCore:code>
      </platformCore:statusDetail>
    </platformCore:status>
  </writeResponse>
</addResponse>
```
Without a zip/postal code it will not be possible to use this address with 3rd Party shippers. Click Cancel to edit the address.

The above customer was added because no error occurred. However, future errors could be avoided by responding appropriately to the warning message.

For a detailed list of all warning messages and the associated codes generated by NetSuite, see Warning Status Codes.

Errors

Errors result if invalid or incomplete data is submitted when performing an operation. For example, if a client attempts to update a customer record with an invalid internal ID, the operation fails and the response contains an error code and message.

When numerous records are submitted within the same request, each is treated individually. For example, if a client attempts to update two events within the same request, where one record has invalid data and the other has valid data, only one of the records has an error and is not updated.

Code

For a detailed list of all error messages and the associated codes, see Error Status Codes.
Error Reporting for afterSubmit Script Failures

When an afterSubmit script runs after a SOAP web services add or update request, and the script fails, the records are still created or updated in NetSuite. The SOAP response indicates that the record has been created or updated even if an afterSubmit script that runs after the SOAP web services request fails.

Using the internal ID of the record, available in the SOAP response, you can avoid creating duplicate records. If the SOAP response indicates that the records have been saved, you only need to retry the user event script. You should not send the SOAP web services request again.

For information about working with afterSubmit scripts, see the help topics afterSubmit(scriptContext) and SuiteScript 2.0 User Event Script Type.

The following is an example of the error message in the SOAP response.

```xml
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Header>
    <platformMsgs:documentInfo xmlns:platformMsgs="urn:messages_2017_1.platform.webservices.netsuite.com">
      <platformMsgs:nsId>WEBSERVICES_3604360_04202017194513975754607083_e2b69fb6b0ef3</platformMsgs:nsId>
    </platformMsgs:documentInfo>
  </soapenv:Header>
  <soapenv:Body>
    <addResponse xmlns="urn:messages_2017_1.platform.webservices.netsuite.com">
      <writeResponse>
        <platformCore:status isSuccess="false" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com">
          <platformCore:statusDetail type="ERROR">
            <platformCore:aftersubmitFailed>true</platformCore:aftersubmitFailed><platformCore:code>USER_ERROR</platformCore:code>
            <platformCore:message>Please enter value(s) for: Company Name, Subsidiary</platformCore:message>
          </platformCore:statusDetail>
          <baseRef internalId="68" externalId="EID_PO_68" type="purchaseOrder" xsi:type="platformCore:RecordRef" xmlns:platformCore="urn:core_2017_1.platform.webservices.netsuite.com"/>
        </platformCore:status>
      </writeResponse>
    </addResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Faults

Faults are exceptions that are of a more fundamental nature than errors. A key distinction between errors and faults is that a fault prevents any operation within a request from being processed whereas an error prevents only a single operation from succeeding on an individual record.

For example, continuing from the case above, if the user's session has timed out when making the request, neither update for either event record is processed and an invalidSessionFault is returned in the response.

Expand code

```xml
<soapenv:Fault>
  <faultcode>soapenv:Server.userException</faultcode>
```
SOAP uses the detail element to capture the error code through the code element and the error message through the message element. The faultcode and faultstring are automatically populated by the server.

Following is an example of a SOAP fault named InvalidCredentialsFault for an invalid e-mail address that is returned as part of the login operation.

**Code**

```xml
<soapenv:Fault>
  <faultcode>soapenv:Server.userExveption</faultcode>
  <faultstring>You have entered an invalid e-mail address or account number. Please try again.</faultstring>
  <detail>
    <platformFaults:invalidCredentialsFault xmlns:urn:faults.platform_2017_1.platform.webservices.netsuite.com>
      <platformFaults:code>INVALID_LOGIN_CREDENTIALS</platformFaults:code>
      <platformFaults:message>You have entered an invalid e-mail address or account number. Please try again.</platformFaults:message>
    </platformFaults:invalidCredentialsFault>
    <ns1:hostname xmlns:ns1="http://xml.apache.org/axis">partners-java10009.bos.netsuite.com</ns1:hostname>
  </detail>
</soapenv:Fault>
```

For a detailed list of fault messages and the associated codes, see SOAP Fault Status Codes. The list does not contain every SOAP fault status code. Some SOAP faults do not have a special fault code.

**Important:** Unexpected errors return an error ID that can be provided to NetSuite Customer Support to help them isolate the error in your account. The following is an example of the error ID:

```xml
<platformCore:code>UNEXPECTED_ERROR</platformCore:code>
<platformCore:message>An unexpected error occurred. Error ID: fevsjhv41tji2juy3le73</platformCore:message>
```

Error IDs will be especially helpful for async operations, as the ID will indicate the error that occurred during the execution of the job, not the retrieval. The exception is if the async error occurred at the platform-level. In this case, the stack will be attached to the job itself.

**SOAP Faults for Each Operation**

The following table lists the SOAP fault types that can be thrown for each operation.
### Login throws...
- ACCT_TEMP_UNAVAILABLE
- EMAIL_ADDRS_REQD
- INVALID_ACCT
- INVALID_LOGIN_CREDENTIALS
- INVALID_VERSION
- PSWD_REQD
- UNEXPECTED_ERROR
- WS_CONCUR_SESSION_DISALLWD
- WS_FEATURE_REQD
- WS_PERMISSION_REQD

### MapSso throws...
- ACCT_TEMP_UNAVAILABLE
- EMAIL_ADDRS_REQD
- INVALID_ACCT
- INVALID_LOGIN_CREDENTIALS
- INVALID_VERSION
- PSWD_REQD
- UNEXPECTED_ERROR
- WS_CONCUR_SESSION_DISALLWD
- WS_FEATURE_REQD
- WS_PERMISSION_REQD

### Logout throws...
- INVALID_VERSION
- UNEXPECTED_ERROR
- WS_CONCUR_SESSION_DISALLWD
- SESSION_TIMED_OUT
- WS_REQUEST_BLOCKED

### add / update / delete throw...
- INVALID_VERSION
- SESSION_TIMED_OUT
- UNEXPECTED_ERROR
### SOAP Faults for Each Operation

#### add / update / delete throw...
- USER_ERROR
- WS_CONCUR_SESSION_DISALLOWED
- WS_LOG_IN_REQD

#### addList / updateList / deleteList throw...
- INVALID_VERSION
- MAX_RCRDS_EXCEEDED
- SESSION_TIMED_OUT
- UNEXPECTED_ERROR
- USER_ERROR
- WS_CONCUR_SESSION_DISALLOWED
- WS_LOG_IN_REQD

#### getList/getAll throw...
- INVALID_VERSION
- MAX_RCRDS_EXCEEDED
- SESSION_TIMED_OUT
- UNEXPECTED_ERROR
- USER_ERROR
- WS_CONCUR_SESSION_DISALLOWED
- WS_LOG_IN_REQD

#### get throws...
- INVALID_VERSION
- SESSION_TIMED_OUT
- UNEXPECTED_ERROR
- USER_ERROR
- WS_CONCUR_SESSION_DISALLOWED
- WS_LOG_IN_REQD

#### search / searchNext / searchMore throw...
- INVALID_VERSION
- MAX_RCRDS_EXCEEDED
- SESSION_TIMED_OUT
### SOAP Fault Status Codes

The following table defines SOAP fault types and their corresponding codes. For a complete description of faults and how they differ from errors and warnings, refer to SOAP Web Services Warnings, Errors, and Faults.

<table>
<thead>
<tr>
<th>Fault Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InsufficientPermissionFault</td>
<td>This fault is thrown when the client does not have the appropriate permissions to perform an action based on the role under which they are currently logged in. If the client (user) has more than one role, they may need to login again supplying a different role with more permissions.</td>
</tr>
<tr>
<td>InvalidAccountFault</td>
<td>This fault is thrown when the client attempts to login with an invalid account id.</td>
</tr>
<tr>
<td>InvalidPartnerCredentials</td>
<td>Partner ID or password submitted with this request is invalid.</td>
</tr>
<tr>
<td>InvalidRequestIP</td>
<td>Originating IP is not one of the registered IPs from which this Partner request may come.</td>
</tr>
<tr>
<td>InvalidSessionFault</td>
<td>This fault can be thrown when the client’s session has timed out or was terminated as the result of a second SOAP web services client establishing another session. The fault message occurs on the first request attempted after the session is terminated/timed out. For more details on how NetSuite handles sessions, see Session Management for SOAP Web Services.</td>
</tr>
<tr>
<td>InvalidVersionFault</td>
<td>This fault is thrown in the event that the request message contains an unsupported version of the schema.</td>
</tr>
<tr>
<td>ExceededConcurrentRequestLimitFault</td>
<td>This fault is thrown in the event that the maximum number of concurrent requests has been reached.</td>
</tr>
<tr>
<td>ExceededRecordCountFault</td>
<td>This fault is thrown in the event the maximum number of records allowed for an operation has been exceeded. For more information, see SOAP Web Services Governance Overview.</td>
</tr>
<tr>
<td>ExceededRequestLimitFault</td>
<td>This fault is thrown if the allowed number of concurrent requests is exceeded. For more information, see SOAP Web Services Governance Overview.</td>
</tr>
<tr>
<td>ExceededRequestSizeFault</td>
<td></td>
</tr>
<tr>
<td>ExceededUsageLimitFault</td>
<td></td>
</tr>
<tr>
<td>UnexpectedErrorFault</td>
<td>This fault is thrown in the event of an occurrence of an unexpected exception.</td>
</tr>
</tbody>
</table>
### Fault Name

<table>
<thead>
<tr>
<th>Fault Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>InvalidCredentialsFault</td>
<td>This fault is thrown in the event of an invalid username (e-mail), password, role, or application ID. This fault is also thrown if two-factor authentication is required but is missing.</td>
</tr>
<tr>
<td>AsyncFault</td>
<td></td>
</tr>
</tbody>
</table>

### Error Status Codes

The following table lists error status code types that can be returned in a message Response. These are values that are used in the code field of the statusDetail type where the type attribute has a value of "error." These codes are listed in the StatusDetailCodeType enumeration, which is defined in the platform faults XSD. You can also view error codes on the Status Detail Code page of the SOAP Schema Browser.

<table>
<thead>
<tr>
<th>Error Code Returned</th>
<th>Long Description or Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABORT_SEARCH_EXCEEDED_MAX_TIME</td>
<td>This search has timed out. You can choose to schedule it to run in the background and have the results emailed to you when complete. On the saved search form, click the &quot;Email&quot; tab, check &quot;Send According to Schedule&quot;, choose an email address on the &quot;Specific Recipients&quot; subtab and a recurrence pattern on the &quot;Schedule&quot; subtab.</td>
</tr>
<tr>
<td>ABORT_UPLOAD_VIRUS_DETECTED</td>
<td>The file {1} contains a virus (2). Upload abort.</td>
</tr>
<tr>
<td>ACCESS_DENIED</td>
<td>Access to this configuration is denied. Please contact NetSuite to gain access to this configuration.</td>
</tr>
<tr>
<td>ACCTNG_PRD_REQD</td>
<td>Missing next accounting period</td>
</tr>
<tr>
<td>ACCT_DISABLED</td>
<td>account disabled</td>
</tr>
<tr>
<td>ACCT_NAME_REQQ</td>
<td>Accounts require a name.</td>
</tr>
<tr>
<td>ACCT_NEEDS_CAMPAIGN_PROVISION</td>
<td>Please contact your account representative to provision campaign emailing for your account.</td>
</tr>
<tr>
<td>ACCT_NOT_CREATED</td>
<td>Account creation was unsuccessful. NetSuite Customer Support has been alerted to this problem.</td>
</tr>
<tr>
<td>ACCT_NUM_REQQ</td>
<td>Missing Account Number. Account number is a required field and it cannot be null or empty.</td>
</tr>
<tr>
<td>ACCT_NUMS_REQQD_OR_DONT_MATCH</td>
<td>Missing ACCT # or ACCT numbers don't match</td>
</tr>
<tr>
<td>ACCT_PERIOD_SETUP_REQQ</td>
<td>The accounting period range {1} has not been defined. Please visit &quot;&lt;A href='{2}'&gt;Setup &gt; Accounting &gt; Manage Accounting Periods&lt;/A&gt;&quot; to define this period or set up your year.</td>
</tr>
<tr>
<td>ACCT_PRDS_BEING_ADDED</td>
<td>Periods are currently being added to this account. Please try again later.</td>
</tr>
<tr>
<td>ACCT_REQD</td>
<td>Attempting to adjust provisioning for a customer without an existing account</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ACCT_TEMP_DISABLED</td>
<td>You have entered an invalid password on (1) consecutive attempts. Access to your account has been suspended for (2) minutes. If you have forgotten your password, please contact Customer Support.</td>
</tr>
<tr>
<td>ACCT_TEMP_UNAVAILABLE</td>
<td>Can't update information - this company's database is currently offline for maintenance. Please try again later.</td>
</tr>
<tr>
<td>ACCT_TEMP_UNAVAILABLE</td>
<td>(Temporarily unavailable)</td>
</tr>
<tr>
<td>ACCT_TEMP_UNAVAILABLE</td>
<td>The account you are trying to access is currently unavailable while we undergo our regularly scheduled maintenance.</td>
</tr>
<tr>
<td>ACCT_TEMP_UNAVAILABLE</td>
<td>We are currently performing maintenance on our system. Please try again soon.</td>
</tr>
<tr>
<td>ACCT_TEMP_UNAVAILABLE</td>
<td>The account you are trying to access is currently unavailable while we undergo our regularly scheduled maintenance.</td>
</tr>
<tr>
<td>ACCT_TEMP_UNAVAILABLE</td>
<td>Your account is disabled for {1} more minutes due to {2} consecutive failed login attempts.</td>
</tr>
<tr>
<td>ACCT_TEMP_UNAVAILABLE</td>
<td>Your account is not yet ready for you to log in. Please wait and try again.</td>
</tr>
<tr>
<td>ACCT_TEMP_UNAVAILABLE</td>
<td>Your company database is offline.</td>
</tr>
<tr>
<td>ACCT_TEMP_UNAVAILABLE</td>
<td>Your data is still being loaded. Please try again later. Contact &lt;a href='app/crm/support/nicorp/support.nl/type=bug&amp;spf=31'&gt;Professional Services&lt;/a&gt; if you have questions.</td>
</tr>
<tr>
<td>ACH_NOT_AVAILBL</td>
<td>ACH Processing is not available in this environment.</td>
</tr>
<tr>
<td>ACH_SETUP_REQD</td>
<td>Account (1) is not setup for ACH transactions.</td>
</tr>
<tr>
<td>ACTIVE_AP_ACCT_REQD</td>
<td>This transaction requires an active Accounts Payable account. Please enable an existing Accounts Payable account, create a new Accounts Payable account, or contact your System Administrator.</td>
</tr>
<tr>
<td>ACTIVE_ROLE_REQD</td>
<td>You can only set an active login role as the SOAP web services default role.</td>
</tr>
<tr>
<td>ACTIVE_TRANS_EXIST</td>
<td>There are active direct deposit transactions for this paycheck.</td>
</tr>
<tr>
<td>ADDITIONAL_AUTHENTICATION_REQUIRED_2FA</td>
<td>Additional Authentication required - 2FA.</td>
</tr>
<tr>
<td>ADDITIONAL_AUTHENTICATION_REQUIRED_SQ</td>
<td>Additional Authentication required - Security Questions.</td>
</tr>
<tr>
<td>ADDRESS_LINE_1_REQD</td>
<td>Address Line 1 is a required field and it cannot be null or empty.</td>
</tr>
<tr>
<td>ADMIN_ACCESS_REQ</td>
<td>At least one active administrator for each account must have access.</td>
</tr>
<tr>
<td>ADMIN_ACCESS_REQ</td>
<td>At least one active administrator for this account must have access.</td>
</tr>
<tr>
<td>ADMIN_ACCESS_REQD</td>
<td>Only administrators may enter a memorized transaction in a closed period.</td>
</tr>
<tr>
<td>ADMIN_ONLY_ACCESS</td>
<td>{1} only the administrator may access this page.</td>
</tr>
<tr>
<td>ADMIN_ONLY_ACCESS</td>
<td>{1} only the administrator may currently access this page.</td>
</tr>
<tr>
<td>ADMIN_USER_REQD</td>
<td>User is not an Admin of the demo account</td>
</tr>
<tr>
<td>ADMISSIBILITY_PACKG_TYP_REQD</td>
<td>An Admissibility Package Type is required for this international shipment.</td>
</tr>
<tr>
<td>ALL_DATA_DELETE_REQD</td>
<td>You must first delete all the data in your account before performing this action. Click &lt;a href='pages/setup/clearaccount.jsp?import=T'&gt;here&lt;/a&gt; to delete your data.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ALL_MTRX_SUBITMES_OPTNS_REQD</td>
<td>The following matrix subitems exist but aren’t included in the options you just specified. On the Matrix tab, please make sure the options you select include all existing subitems:&lt;p&gt; (1)</td>
</tr>
<tr>
<td>ALREADY_IN_INVT</td>
<td>The following (1) numbers are already in inventory: (2)</td>
</tr>
<tr>
<td>ALREADY_IN_INVT</td>
<td>The following (1) number is already in inventory: (2)</td>
</tr>
<tr>
<td>AMORTZN_INVALID_DATE_RANGE</td>
<td>Amortization end date can not be before amortization start date.</td>
</tr>
<tr>
<td>AMORTZN_TMPLT_DATA_MISSING</td>
<td>One or more line items on this transaction have Variable Amortization Templates, but do not have the required (1) also populated. Please either change the Template for these items or indicate which (1) will be used to schedule the amortization.</td>
</tr>
<tr>
<td>AMT_DISALLWD</td>
<td>Description items may not have an amount.</td>
</tr>
<tr>
<td>AMT_EXCEEDS_APPROVAL_LIMIT</td>
<td>No one in your chain of command has a sufficient spending limit to approve this transaction.</td>
</tr>
<tr>
<td>ANSWER_REQD</td>
<td>Please provide an answer.</td>
</tr>
<tr>
<td>APPROVAL_PERMS_REQD</td>
<td>(1) The restrictions on your role do not allow you to approve or reject this record.</td>
</tr>
<tr>
<td>AREA_CODE_REQD</td>
<td>Please include an area code with the phone number.</td>
</tr>
<tr>
<td>ASSIGNEE_REQD</td>
<td>(1) must be assigned to (2)</td>
</tr>
<tr>
<td>AT_LEAST_ONE_FILE_REQD</td>
<td>Download folder must have at least one file.</td>
</tr>
<tr>
<td>AT_LEAST_ONE_PACKAGE_REQD</td>
<td>1 or more packages are required.</td>
</tr>
<tr>
<td>AT_LEAST_ONE_RETURN_FLD_REQD</td>
<td>You must specify at least one field to be returned.</td>
</tr>
<tr>
<td>AT_LEAST_ONE_SUB_REQD</td>
<td>You must choose at least one subsidiary.</td>
</tr>
<tr>
<td>ATTACH_SIZE_EXCEEDED</td>
<td>The data you are uploading exceeds the maximum allowable size of (1). Please change your selection and try again</td>
</tr>
<tr>
<td>ATTACH_SIZE_EXCEEDED</td>
<td>This message exceeds the limit of 24 MB. Please reduce the size of the message and its attachments and try again.&lt;br/&gt;Note: Files can be larger when attached due to encoding.</td>
</tr>
<tr>
<td>ATTACHMNT_CONTAINS_VIRUS</td>
<td>The attachment with file name (1) contains a virus (2). It is removed from the message.</td>
</tr>
<tr>
<td>ATTACHMNT_CONTAINS_VIRUS</td>
<td>The attachment file (0) contains virus (1). Save message abort.</td>
</tr>
<tr>
<td>AUDIT_W2_1099</td>
<td>Use this report to audit the information that will be used to generate W2s and 1099s.</td>
</tr>
<tr>
<td>AUTO_NUM_UPDATE_DISALLWD</td>
<td>We currently do not support an automatic numbering update of more than (1) (2) records. Please contact &lt;a href='<a href="http://app/crm/support/nlcorpssupport.nl?type=support'%3ENetSuite">http://app/crm/support/nlcorpssupport.nl?type=support'&gt;NetSuite</a> Customer Support&lt;/a&gt; to request a full numbering update of your (2)s.&lt;/a&gt;</td>
</tr>
<tr>
<td>AVS_ERROR</td>
<td>This order failed the AVS check and will not be saved.&lt;br&gt;&lt;br&gt;An AVS mismatch has been detected and the transaction has been rejected. However, a valid authorization has been created against the cardholder’s account. If you want to accept this order, click Go Back and check the Ignore AVS checkbox and Save the order. Note that this will result in a second authorization.</td>
</tr>
<tr>
<td>BALANCE_EXCEEDS_CREDIT_LIMIT</td>
<td>Customer balance exceeds credit limit</td>
</tr>
<tr>
<td>BANK_ACCT_REQD</td>
<td>You must have a bank account to perform this operation. Click &lt;a href='<a href="http://app/accounting/account/account.nl'&amp;gt;here&amp;lt;/a">http://app/accounting/account/account.nl'&amp;gt;here&amp;lt;/a</a>&gt; to add one.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>BASE_CRNCY_REQD</td>
<td>You may not delete your base currency.</td>
</tr>
<tr>
<td>BILL_PAY_STATUS_UNAVAILABLE</td>
<td>View Online Bill Pay Status information is currently not available. Please try again in a few minutes.</td>
</tr>
<tr>
<td>BILL_PAY_STATUS_UNAVAILABLE</td>
<td>View Online Bill Pay Status is not available until your bill pay registration is complete.</td>
</tr>
<tr>
<td>BILL_PMTS_MADE_FROM_ACCT_ONLY</td>
<td>Your payment has been recorded, but online bill payments can only be made from the account <code>{1}</code>, so no online bill pay will be made. You should return to the payment screen if you wish to print the check.</td>
</tr>
<tr>
<td>BILLABLES_DISALLOWD</td>
<td><code>{1}</code> does not allow billables.</td>
</tr>
<tr>
<td>BILLING_ISSUES</td>
<td>Your account has been locked due to billing issues. You must call your NetSuite Sales Representative for further assistance.</td>
</tr>
<tr>
<td>BILLING_ISSUES</td>
<td>Your account has not been fully paid for. Please log in to your account and follow the billing process or contact your Account Manager.</td>
</tr>
<tr>
<td>BILLING_SCHEDUL_INVALID_RECURR</td>
<td>Billing schedules may not have a recurrence count greater than 500.</td>
</tr>
<tr>
<td>BILLPAY_APPROVAL_UNAVAILBL</td>
<td>Approve Online Bill Payments is currently not available. Please try again in a few minutes.</td>
</tr>
<tr>
<td>BILLPAY_REGISTRATION_REQD</td>
<td>Online bill pay approve payments is not available until your billpay registration is complete.</td>
</tr>
<tr>
<td>BILLPAY_SRVCR_UNAVAILBL</td>
<td>Online Bill Pay service is temporarily suspended. If you prefer to wait until Online Bill Pay service is restored, just leave payments to be approved in this list. We will notify you by email as soon as the service is available again. If you need to make an urgent payment, we suggest that you print a check. To do this: Clear the Online Bill Pay check box for that payment. Click the underlined date of the payment, and you will return to the Bill Payment page. Go to Transactions &gt; Print Checks and Forms &gt; Checks, and then mark the check to be printed. Click Submit.</td>
</tr>
<tr>
<td>BIN_DSNT_CONTAIN_ENOUGH_ITEM</td>
<td>The following bins do not contain enough of the requested item <code>{1}</code>: <code>{2}</code></td>
</tr>
<tr>
<td>BIN_DSNT_CONTAIN_ENOUGH_ITEM</td>
<td>The following bin does not contain enough of the requested item <code>{1}</code>: <code>{2}</code></td>
</tr>
<tr>
<td>BIN_ITEM_UNAVAILBL</td>
<td>The following bins are not available for the specified item: <code>{1}</code></td>
</tr>
<tr>
<td>BIN_ITEM_UNAVAILBL</td>
<td>The following bins are not available for the specified item: <code>{1}</code></td>
</tr>
<tr>
<td>BIN_ITEM_UNAVAILBL</td>
<td>The following bin is not available for the specified item: <code>{1}</code></td>
</tr>
<tr>
<td>BIN_ITEM_UNAVAILBL</td>
<td>The following bin is not available for the specified item: <code>{1}</code></td>
</tr>
<tr>
<td>BIN_SETUP_REQD</td>
<td>The following bins are not associated with the item: <code>{1}</code>: <code>{2}</code>. You can associate bins with an item on the inventory tab of the item record.</td>
</tr>
<tr>
<td>BIN_UNDEFND</td>
<td>The following bins specified for the item <code>{1}</code> are not defined in the transaction location <code>{2}</code>: <code>{3}</code></td>
</tr>
<tr>
<td>BUNDLE_IS_DEPRECATED</td>
<td>This bundle is no longer available. It has been deprecated by bundle <code>{1}</code> on account <code>{2}</code>.</td>
</tr>
<tr>
<td>BUNDLE_IS_DEPRECATED</td>
<td>You cannot update this bundle as it has been deprecated and you have not been granted access to the replacement bundle. Please contact the solution provider.</td>
</tr>
<tr>
<td>CALENDAR_PREFS_REQD</td>
<td>Set up <code>{1}</code> Calendar Preferences first.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CALENDAR_PREFS_REQD</td>
<td>Set up Calendar Preferences first</td>
</tr>
<tr>
<td>CAMPAGIN_ALREADY_EXECUTED</td>
<td>You cannot delete email campaigns that have already been executed</td>
</tr>
<tr>
<td>CAMPAIGN_IN_USE</td>
<td>You cannot delete a campaign event that already has activity.</td>
</tr>
<tr>
<td>CAMPAIGN_SET_UP_REQD</td>
<td>The following steps need to be performed before a campaign can be created:&lt;p&gt;(1)</td>
</tr>
<tr>
<td>CANNOT_RESET_PASSWORD</td>
<td>Sorry, we are currently unable to reset your password. Please contact support or try again later.</td>
</tr>
<tr>
<td>CANT_APPLY_PMT</td>
<td>The top level entity cannot accept payment because it has a status of {1}. It must have a status of {2} or {3} to accept payment.</td>
</tr>
<tr>
<td>CANT_APPLY_PMT</td>
<td>This entity cannot accept payment because it has a status of {1}. It must have a status of {2} or {3} to accept payment.</td>
</tr>
<tr>
<td>CANT_AUTO_CREATE_ADJSTMNT</td>
<td>The ‘Intercompany Expenses’ accounting preference does not currently support the automated creation of adjustments. If you would like to create adjustments automatically, please change that preference to ‘Allow and Auto Adjust.’</td>
</tr>
<tr>
<td>CANT_CALC_FEDEX_RATES</td>
<td>FedEx rates cannot be calculated:</td>
</tr>
<tr>
<td>CANT_CANCEL_APPRVD_RETAIN_AUTH</td>
<td>You cannot cancel this return authorization because it has already been approved.</td>
</tr>
<tr>
<td>CANT_CANCEL_BILL_PMT</td>
<td>The Online Bill Payment cannot be stopped because the payment may already have been made.</td>
</tr>
<tr>
<td>CANT_CHANGE_COMMSSN</td>
<td>The Online Bill Payment cannot be stopped because the payment may already have been made.</td>
</tr>
<tr>
<td>CANT_CHANGE_CONTACT_RESTRICTN</td>
<td>You cannot change the restriction on this contact.</td>
</tr>
<tr>
<td>CANT_CHANGE_CRMRECORDTYPELINKS</td>
<td>Cannot alter standard CrmRecordTypeLinks</td>
</tr>
<tr>
<td>CANT_CHANGE_EVENT_PRIMARY_TYP</td>
<td>You cannot change the primary type for this event</td>
</tr>
<tr>
<td>CANT_CHANGE_IP_ADDRESS</td>
<td>The domain {1} is currently associated with IP address {2}. You cannot change the IP address of a live domain.</td>
</tr>
<tr>
<td>CANT_CHANGE_LEAD_SOURCE_CAT</td>
<td>You cannot change the category for a leadsource that is defined as the default leadsource for another category</td>
</tr>
<tr>
<td>CANT_CHANGE_PSWD</td>
<td>You changed your password less than 24 hours ago. NetSuite only allows one password change per 24-hour period.</td>
</tr>
<tr>
<td>CANT_CHANGE_REV_REC_TMPLT</td>
<td>The rev rec template on a billable expense can not be changed or removed after it is saved.</td>
</tr>
<tr>
<td>CANT_CHANGE_REV_REC_TMPLT</td>
<td>The rev rec template on billable time and items can not be changed or removed after it is saved.</td>
</tr>
<tr>
<td>CANT_CHANGE_SUB</td>
<td>You cannot change the subsidiary on this record because doing so will change the subsidiary selected on the associated employee record.</td>
</tr>
<tr>
<td>CANT_CHANGE_TASK_LINK</td>
<td>Cannot alter standard task links</td>
</tr>
<tr>
<td>CANT_CHANGE_UNITS_TYP</td>
<td>You may not change the units type of an item after it has been set.</td>
</tr>
<tr>
<td>CANT_CHANGE_VSOE_ALLOCTN</td>
<td>You are attempting to change the VSOE Allocation for a transaction in a closed period. You must either change the posting period for the related transaction or open the period.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CANT_CHG_POSTED_BILL_VRNC</td>
<td>The receipts for this bill can not be changed after the bill variance has been posted.</td>
</tr>
<tr>
<td>CANT_CHG_POSTED_BILL_VRNC</td>
<td>One or more lines have had their bill variance posted and can not be changed.</td>
</tr>
<tr>
<td>CANT_COMPLETE_FULFILL</td>
<td>The fulfillment cannot be completed.</td>
</tr>
<tr>
<td>CANT_CONNECT_TO_STORE</td>
<td>Error - Unable to connect to store {1}</td>
</tr>
<tr>
<td>CANT_CONVERT_CLASS_DEPT</td>
<td>You cannot convert classes to departments because you already have department data in the system. You must remove all departments first.</td>
</tr>
<tr>
<td>CANT_CONVERT_CLASS_LOC</td>
<td>You cannot convert classes to locations because you already have location data in the system. You must remove all locations first.</td>
</tr>
<tr>
<td>CANT_CONVERT_INVIT_ITEM</td>
<td>This item is a member of a kit or a non-numbered assembly. You may not convert it to a numbered inventory item.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because the In Bond Code field is not set. Please enter a value in the In Bond Code field on the Item Fulfillment page.</td>
</tr>
<tr>
<td>CANT_CREATE_FILES</td>
<td>Could not create files for uploading your data</td>
</tr>
<tr>
<td>CANT_CREATE_NON_UNIQUE_RCRD</td>
<td>A record with the same unique signatures already exists. You must enter unique signatures for each record you create.</td>
</tr>
<tr>
<td>CANT_CREATE_PO</td>
<td>Purchase Orders cannot be created for assembly items.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because a currency must be defined for the &quot;Ship From&quot; country &quot;{1}&quot; when using the Insured Value option. Go to Lists -&gt; Accounting -&gt; Currencies to create a currency for {1}.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because the Package Weight was not entered. Please enter a value in the Package Weight field on the Item Fulfillment page.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because the Phone Number of the &quot;Ship From&quot; address is not set. Please go to $(regex) to set the &quot;Ship From&quot; Phone Number.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because the Phone Number of the &quot;Ship To&quot; address is not set. Please enter a &quot;Ship To&quot; Phone Number on the Item Fulfillment page.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because the Pickup Type was not set. Please go to Setup &gt; Set Up Shipping to select a shipping Pickup Type.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because the shipping method was not set. Please go to Lists &gt; Shipping Items to select a Shipping Label Integration shipping method for this shipping item.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because the State field of the &quot;Ship To&quot; address is not set. Please enter a &quot;Ship To&quot; State on the Item Fulfillment page.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because the State of the &quot;Ship From&quot; address is not set. Please go to $(regex) to enter the &quot;Ship From&quot; State.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because the Zip Code of the &quot;Ship From&quot; address is not set. Please go to $(regex) to enter the &quot;Ship From&quot; Zip Code.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because your {1} Account Number is not set. Go to Setup &gt; Set Up Shipping &gt; (2) Registration to enter your (3) Account Number.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because your {1} Registration Address Line 1 is not set. Go to Setup &gt; (2) Registration to complete the Address Line 1 field.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because your {1} Registration City is not set. Go to Setup &gt; (2) Registration to enter your City.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because your {1} Registration Company field is not set. Go to Setup &gt; (2) Registration to enter a name in the Company field.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because your {1} Registration Country is not set. Go to Setup &gt; (2) Registration to select your Country.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because your {1} Registration Ship to Attention field is not set. Go to Setup &gt; (2) Registration to enter a name in the Ship to Attention field.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because your {1} Registration State is not set. Go to Setup &gt; (2) Registration to select or enter your State.</td>
</tr>
<tr>
<td>CANT_CREATE_SHIP_LABEL</td>
<td>A shipping label could not be generated because your {1} Registration Zip Code is not set. Go to Setup &gt; (2) Registration to enter your Zip Code.</td>
</tr>
<tr>
<td>CANT_CREATE_WORK_ORD</td>
<td>Work orders can only be created for assembly items.</td>
</tr>
<tr>
<td>CANT_DEL_DEFAULTCALENDAR</td>
<td>Work orders can only be created for assembly items.</td>
</tr>
<tr>
<td>CANT_DEL_DEFAULT_SHIP_METHOD</td>
<td>This Shipping Item cannot be deleted because it is the Default Shipping Method. Please go to Setup &gt; Accounting &gt; Set Up Shipping and choose a new Default Shipping Method before deleting this Shipping Item.</td>
</tr>
<tr>
<td>CANT_DEL_REALIZED_GAINLOSS</td>
<td>A Realized Gain/Loss Transaction cannot be deleted.</td>
</tr>
<tr>
<td>CANT_DEL_TRANS_RVRSL</td>
<td>The reversal of the month-end Unrealized Gain/Loss transaction cannot be deleted.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>CANT_DELETE_ACCT</td>
<td>This account cannot be deleted because it has associated transactions.</td>
</tr>
<tr>
<td>CANT_DELETE_ACCT</td>
<td>This account cannot be deleted because it is a special type of account needed by {1}.</td>
</tr>
<tr>
<td>CANT_DELETE_ACCT</td>
<td>This account cannot be deleted because it is a special type of account needed by NetSuite.</td>
</tr>
<tr>
<td>CANT_DELETE_ACCT</td>
<td>This account cannot be deleted because it is a special type of account needed by the system.</td>
</tr>
<tr>
<td>CANT_DELETE_ACCT</td>
<td>This account cannot be deleted because it is used by one or more transactions or it has child accounts or it is used by one or more items.</td>
</tr>
<tr>
<td>CANT_DELETE.ACCT.PRO</td>
<td>You may not delete an accounting period with transactions posted to it. You must first edit the transactions, change the posting period and then delete the period.</td>
</tr>
<tr>
<td>CANT_DELETE_ALLOCTN</td>
<td>This allocation detail cannot be deleted because it has a journal entry.</td>
</tr>
<tr>
<td>CANT_DELETE_BIN</td>
<td>You may not delete this bin record because it is already in use. You must either remove all references to it in item records and transactions or make it inactive.</td>
</tr>
<tr>
<td>CANT_DELETE_CATEGORY</td>
<td>This category cannot be deleted because it has child items.</td>
</tr>
<tr>
<td>CANT_DELETE_CATEGORY</td>
<td>This category cannot be deleted because it has subcategories.</td>
</tr>
<tr>
<td>CANT_DELETE_CC_PROCESSOR</td>
<td>This credit card processor is used in transaction and cannot be deleted.</td>
</tr>
<tr>
<td>CANT_DELETE_CELL</td>
<td>This cell cannot be deleted because it has child items.</td>
</tr>
<tr>
<td>CANT_DELETE_CHILD_RCRD_FOUND</td>
<td>This (1) record cannot be deleted because it is referenced by other records.</td>
</tr>
<tr>
<td>CANT_DELETE_CHILD_RCRDS_EXIST</td>
<td>This record cannot be deleted because it has child records.</td>
</tr>
<tr>
<td>CANT_DELETE_CLASS</td>
<td>This class cannot be deleted because it has child items.</td>
</tr>
<tr>
<td>CANT_DELETE_COLOR_THEME</td>
<td>This color theme cannot be deleted because it is being used.</td>
</tr>
<tr>
<td>CANT_DELETE_COMMSSN_SCHDUL</td>
<td>This schedule has already been used to generate commission calculations and can’t be deleted. If no authorizations have been made, schedule can be deleted after being removed from all active plans.</td>
</tr>
<tr>
<td>CANT_DELETE_COMPANY</td>
<td>This company cannot be deleted because it has child entities.</td>
</tr>
<tr>
<td>CANT_DELETE_COMPANY_TYP</td>
<td>This company type cannot be deleted because the company has associated transactions.</td>
</tr>
<tr>
<td>CANT_DELETE_CONTACT_HAS_CHILD</td>
<td>The contact record cannot be deleted because it has child records.</td>
</tr>
<tr>
<td>CANT_DELETE_CONTACT_HAS_CHILD</td>
<td>This contact cannot be deleted because it has child entities.</td>
</tr>
<tr>
<td>CANT_DELETE_CSTM_FIELD</td>
<td>This custom field cannot be deleted because it is referred to by other custom fields.</td>
</tr>
<tr>
<td>CANT_DELETE_CSTM_FORM</td>
<td>This custom form cannot be deleted because it is referred to by other custom forms.</td>
</tr>
<tr>
<td>CANT_DELETE_CSTM_ITEM_FIELD</td>
<td>This custom item field has dependent matrix items. It can not be deleted.</td>
</tr>
<tr>
<td>CANT_DELETE_CSTM_LAYOUT</td>
<td>This custom layout cannot be deleted because it is used by custom forms.</td>
</tr>
<tr>
<td>CANT_DELETE_CSTM_LIST</td>
<td>This custom list cannot be deleted because it is referred to by custom fields.</td>
</tr>
<tr>
<td>CANT_DELETE_CSTM_RCRD</td>
<td>This custom record cannot be deleted because it is referred to by custom fields.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CANT_DELETE_CSTM_RCRD_ENTRY</td>
<td>This custom record entry cannot be deleted because it is referred to by other records</td>
</tr>
<tr>
<td>CANT_DELETE_CUST</td>
<td>You can't delete this customer because it's set up as default Anonymous Customer</td>
</tr>
<tr>
<td>CANT_DELETE_CUSTOMER</td>
<td>This customer or job cannot be deleted because it has child entities.</td>
</tr>
<tr>
<td>CANT_DELETE_DEFAULT_FLDR</td>
<td>You cannot delete the default folders.</td>
</tr>
<tr>
<td>CANT_DELETE_DEFAULT_PRIORITY</td>
<td>You cannot delete the default case priority. Please select a new default first.</td>
</tr>
<tr>
<td>CANT_DELETE_DEFAULT_SALES_REP</td>
<td>Default Sales Rep Role cannot be deleted.</td>
</tr>
<tr>
<td>CANT_DELETE_DEFAULT_STATUS</td>
<td>You cannot delete a default case status. Please select a new default first.</td>
</tr>
<tr>
<td>CANT_DELETE_DEFAULT_STATUS</td>
<td>You can't delete or inactivate that status because it is a set up as a default status. Please navigate to &lt;a href='/app/setup/sfasetup.nl' target='_blank'&gt;Sales Preferences&lt;/a&gt; and change that status</td>
</tr>
<tr>
<td>CANT_DELETE_DEFAULT_VALUE</td>
<td>You may not delete or inactivate that value because it is a default. Please select a new default first.</td>
</tr>
<tr>
<td>CANT_DELETE_DEFAULT_WEBSITE</td>
<td>The default Web site cannot be deleted.</td>
</tr>
<tr>
<td>CANT_DELETE_EMPL</td>
<td>This employee cannot be deleted because it has child entities</td>
</tr>
<tr>
<td>CANT_DELETE_ENTITY</td>
<td>This entity cannot be deleted because it has child items</td>
</tr>
<tr>
<td>CANT_DELETE_FIN_STATMNT_LAYOUT</td>
<td>This financial statement layout cannot be deleted because it is referred to by other layouts.</td>
</tr>
<tr>
<td>CANT_DELETE_FLDR</td>
<td>These predefined folders cannot be deleted</td>
</tr>
<tr>
<td>CANT_DELETE_HAS_CHILD_ITEM</td>
<td>This (1) cannot be deleted because it has child items</td>
</tr>
<tr>
<td>CANT_DELETE_INFO_ITEM</td>
<td>This information item cannot be deleted because it has child items</td>
</tr>
<tr>
<td>CANT_DELETE_ITEM</td>
<td>This item cannot be deleted because it has child items</td>
</tr>
<tr>
<td>CANT_DELETE_ITEM_LAYOUT</td>
<td>This item/category layout cannot be deleted because it is used by store tabs</td>
</tr>
<tr>
<td>CANT_DELETE_ITEM_TMPLT</td>
<td>This item/category template cannot be deleted because it is referred to by a theme or an item</td>
</tr>
<tr>
<td>CANT_DELETE_JOB_RESOURCE_ROLE</td>
<td>Default Job Resource Role cannot be deleted</td>
</tr>
<tr>
<td>CANT_DELETE_LEGACY_CATEGORY</td>
<td>Legacy category cannot be removed</td>
</tr>
<tr>
<td>CANT_DELETE_LINE</td>
<td>This line cannot be deleted, because it is referred to by other records. Before removing this line, remove any discount or markup lines applied to it.</td>
</tr>
<tr>
<td>CANT_DELETE_MEDIA_ITEM</td>
<td>This media item cannot be deleted because it is being referenced by another item.</td>
</tr>
<tr>
<td>CANT_DELETE_MEMRZD_TRANS</td>
<td>This memorized transaction cannot be deleted because it referenced in transactions</td>
</tr>
<tr>
<td>CANT_DELETE_OR_CHANGE_ACCT</td>
<td>Special accounts cannot be deleted and their type cannot be changed</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>CANT_DELETE_PLAN_ASSGNMNT</td>
<td>Trying to delete plan assignment referenced by precalc.</td>
</tr>
<tr>
<td>CANT_DELETE_PRESNTN_CAT</td>
<td>This presentation category cannot be deleted because it has subcategories.</td>
</tr>
<tr>
<td>CANT_DELETE_RCRD</td>
<td>This (1) record cannot be deleted because it referenced by other records.</td>
</tr>
<tr>
<td>CANT_DELETE_RCRD</td>
<td>This record cannot be deleted because it has (1) child records.</td>
</tr>
<tr>
<td>CANT_DELETE_RCRD</td>
<td>This record cannot be deleted because it is referenced by other records or it is used by one or more transactions.</td>
</tr>
<tr>
<td>CANT_DELETE_RCRD</td>
<td>This record cannot be deleted, because it is referred to by other records.</td>
</tr>
<tr>
<td>CANT_DELETE_RCRDS</td>
<td>Selected records could not be deleted because one or more of them are of a special type of account needed by (1).</td>
</tr>
<tr>
<td>CANT_DELETE_RCRDS</td>
<td>Selected records could not be deleted because one or more of them are referenced by other records.</td>
</tr>
<tr>
<td>CANT_DELETE_SITE_TAG</td>
<td>This site tag cannot be deleted because it is being used.</td>
</tr>
<tr>
<td>CANT_DELETE_SITE_THEME</td>
<td>This site theme cannot be deleted because it is being used.</td>
</tr>
<tr>
<td>CANT_DELETE_SOLUTN</td>
<td>This solution cannot be deleted because it has been applied to support cases.</td>
</tr>
<tr>
<td>CANT_DELETE_STATUS_TYPE</td>
<td>You cannot delete the only status of type (1).</td>
</tr>
<tr>
<td>CANT_DELETE_SUBTAB</td>
<td>This subtab cannot be deleted because it is referred to by custom fields.</td>
</tr>
<tr>
<td>CANT_DELETE_SYSTEM_NOTE</td>
<td>You cannot alter or delete a system logged note.</td>
</tr>
<tr>
<td>CANT_DELETE_TAX_VENDOR</td>
<td>This is a special tax vendor and cannot be deleted.</td>
</tr>
<tr>
<td>CANT_DELETE_TMPLT_RCRD</td>
<td>This template record cannot be deleted.</td>
</tr>
<tr>
<td>CANT_DELETE_TRANS</td>
<td>This transaction cannot be deleted because it is a Google Checkout order awaiting updated payment information.</td>
</tr>
<tr>
<td>CANT_DELETE_TRANS</td>
<td>This transaction cannot be deleted because it is a posting Google Checkout order.</td>
</tr>
<tr>
<td>CANT_DELETE_TRANS</td>
<td>This transaction cannot be deleted because it is linked to one or more commission transactions. The commission authorizations due to this transaction need to be removed to be able to delete this transaction.</td>
</tr>
<tr>
<td>CANT_DELETE_TRANS</td>
<td>This transaction cannot be deleted because it is referenced by an intercompany adjustment. The adjustment must be deleted first.</td>
</tr>
<tr>
<td>CANT_DELETE_TRANS</td>
<td>This transaction cannot be deleted because it is referred to by other transactions. It may be a bill or an invoice that has been paid or an expense that has been reimbursed.</td>
</tr>
<tr>
<td>CANT_DELETE_TRAN_LINE</td>
<td>Failed to delete line (1). This line is linked to another transaction.</td>
</tr>
<tr>
<td>CANT_DELETE_TRAN_LINES</td>
<td>Lines with partially recognized rev rec or amortization schedules cannot be deleted.</td>
</tr>
<tr>
<td>CANT_DELETE_UPDATE_ACCT</td>
<td>This account cannot be deleted or changed because it is a special type of account needed by (1).</td>
</tr>
<tr>
<td>CANT_DELETE_URL</td>
<td>This third party conversion tracking URL cannot be deleted because it is reference by other records.</td>
</tr>
<tr>
<td>CANT_DELETE_VENDOR</td>
<td>This (1:Vendor) is related to a (2:Partner) which is eligible for commission and cannot be deleted.</td>
</tr>
<tr>
<td>CANT_DELETE_VENDOR</td>
<td>This vendor cannot be deleted because there are dependent items, such as a pending payment. If you wish to remove the payee, you must first delete all such dependent items.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CANT_DIVIDE_BY_ZERO</td>
<td>There is a divide by zero error in this search. It may be an error with a formula you have</td>
</tr>
<tr>
<td></td>
<td>used. Please retry without the formula(s). If an error still occurs, please file with</td>
</tr>
<tr>
<td></td>
<td>Customer Support. If it does not, please correct your formula. Typically that consists of</td>
</tr>
<tr>
<td></td>
<td>taking the denominator and wrapping it in <code>NULLIF(&lt;denominator&gt;,0)</code></td>
</tr>
<tr>
<td>CANT_DOWNLOAD_EXPIRED_FILE</td>
<td>This file has expired and can no longer be downloaded</td>
</tr>
<tr>
<td>CANT_EDIT_CHARGED_ORDER</td>
<td>You cannot charge an order that is already completely charged.</td>
</tr>
<tr>
<td>CANT_EDIT_CUST_LIST</td>
<td>Can not modify this Custom List because its entries are in use.</td>
</tr>
<tr>
<td>CANT_EDIT_CUST_PMT</td>
<td>This customer payment cannot be edited while it has an Automated Clearing House transmission</td>
</tr>
<tr>
<td></td>
<td>in process.</td>
</tr>
<tr>
<td></td>
<td>To view the status of customer payments with ACH transmissions, go to Transactions &gt;</td>
</tr>
<tr>
<td></td>
<td>View Electronic Funds Transfer Status.</td>
</tr>
<tr>
<td>CANT_EDIT_DPLYMNT_IN_PROG</td>
<td>You cannot change or delete a deployment that is in progress or in the queue.</td>
</tr>
<tr>
<td>CANT_EDIT_DPLYMNT_IN_PROG</td>
<td>You cannot edit a script deployment when it is being executed.</td>
</tr>
<tr>
<td>CANT_EDIT_FOLDER</td>
<td>Predefined folders cannot be updated.</td>
</tr>
<tr>
<td>CANT_EDIT_OLD_CASE</td>
<td>This case cannot be edited because it was closed (1) or more days ago.</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot alter standard dashboards</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot alter standard dashboard role maps</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot alter standard dashboard section maps</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot alter standard fields</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot alter standard forms or layouts</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot alter standard portlets</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot alter standard report snapshot layouts</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot alter standard roles</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot alter standard searches</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot alter standard sections</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot alter standard tabs</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot alter standard tasks</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot alter standard task categories</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot alter standard templates</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot Alter Standard Types</td>
</tr>
<tr>
<td>CANT_EDIT_STANDARD_OBJ</td>
<td>Cannot alter standard words</td>
</tr>
<tr>
<td>CANT_EDIT_TAGATA</td>
<td>The Receivable Tegata is linked to Invoices and is no longer editable</td>
</tr>
<tr>
<td>CANT_EDIT_TRAN</td>
<td>You cannot edit intercompany adjustments.</td>
</tr>
<tr>
<td>CANT_EDIT_TRAN</td>
<td>You cannot edit the account of a transaction line that is linked to others</td>
</tr>
<tr>
<td>CANT_EDIT_TRAN</td>
<td>You cannot edit this expense report</td>
</tr>
</tbody>
</table>

**SuiteTalk SOAP Web Services Platform Guide**
<table>
<thead>
<tr>
<th>Error Code Returned</th>
<th>Long Description or Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANT_EDIT_TRAN</td>
<td>You cannot edit this transaction because it was automatically created by the Payroll feature.</td>
</tr>
<tr>
<td>CANT_ESTABLISH_LINK</td>
<td>Unable to establish link with {1}</td>
</tr>
<tr>
<td>CANT_FIND_BUG</td>
<td>Cannot locate the bug that was just entered (1)!</td>
</tr>
<tr>
<td>CANT_FIND_MAIL_MERGE_ID</td>
<td>Mail Merge Id not found</td>
</tr>
<tr>
<td>CANT_FIND_RCRD</td>
<td>Could not find record with {1} = {2}</td>
</tr>
<tr>
<td>CANT_FIND_SAVED_IMPORT</td>
<td>No saved import with internalId {1}</td>
</tr>
<tr>
<td>CANT_FIND_SOURCE_AMORTZN_ACCT</td>
<td>The source account for the amortization schedule could not be determined.</td>
</tr>
<tr>
<td>CANT_FIND_UPS_REG_FOR_LOC</td>
<td>No UPS registration was found for the location selected. Please select a different shipping</td>
</tr>
<tr>
<td>CANT_FULFILL_ITEM</td>
<td>An item receipt has been posted</td>
</tr>
<tr>
<td>CANT_INACTIVATE_COMMSN_PLAN</td>
<td>You cannot inactivate a plan that has commission payments that are pending authorization.</td>
</tr>
<tr>
<td></td>
<td>Please clear the commission payments at Transactions &gt; Authorize Commissions before</td>
</tr>
<tr>
<td></td>
<td>inactivating this plan.</td>
</tr>
<tr>
<td>CANT_INACTIVE_DEFAULT_SYNC_CAT</td>
<td>You cannot inactivate the default synchronization category.</td>
</tr>
<tr>
<td>CANT_INACTIVE_DEFAULT_TMPLT</td>
<td>You cannot inactivate this template record because it is set up as a default template</td>
</tr>
<tr>
<td>CANT_LOAD_SAVED_SEARCH_PARAM</td>
<td>Error loading saved search params</td>
</tr>
<tr>
<td>CANT_LOGIN_WITH_OAUTH</td>
<td>A login operation or Request Level Credentials must not be used in conjunction with OAuth</td>
</tr>
<tr>
<td></td>
<td>authorization (do not translate 'login' or capitalized words)</td>
</tr>
<tr>
<td>CANT_MAKE_CONTACT_PRIVATE</td>
<td>Employee contacts cannot be made private</td>
</tr>
<tr>
<td>CANT_MAKE_CONTACT_PRIVATE</td>
<td>Individual relationship contacts cannot be made private</td>
</tr>
<tr>
<td>CANT_MARK_SHIPPED</td>
<td>Item {1:item name} cannot be marked shipped because the remaining quantity on the</td>
</tr>
<tr>
<td></td>
<td>linked purchase order does not match the remaining quantity on the sales order.</td>
</tr>
<tr>
<td>CANT_MERGE_EMPLS</td>
<td>Employees can not be merged</td>
</tr>
<tr>
<td>CANT_MODIFY_APPROVD_TIME</td>
<td>Time records can not be modified after they have been approved.</td>
</tr>
<tr>
<td>CANT_MODIFY_FULFILL_STATUS</td>
<td>You may not change the fulfillable status of an item which has transactions associated with it.</td>
</tr>
<tr>
<td>CANT_MODIFY_ISSUE_STATUS</td>
<td>The issue status '{1:issue status name}' cannot be changed from {2:base status} to {3:base status} because it is in use.</td>
</tr>
<tr>
<td>CANT_MODIFY_LOCKED_FLD</td>
<td>You may not update or delete a locked custom field.</td>
</tr>
<tr>
<td>CANT_MODIFY_PARENT</td>
<td>Payments have been accepted from the top level parent. The top level parent can not be</td>
</tr>
<tr>
<td></td>
<td>changed.</td>
</tr>
<tr>
<td>CANT_MODIFY_REV_REC</td>
<td>The value of Rev Rec on Rev Commit may not be modified for this transaction.</td>
</tr>
<tr>
<td>CANT_MODIFY_SUB</td>
<td>You cannot change the subsidiary of this entity because one or more transactions exist for this entity.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>CANT_MODIFY_TAGATA</td>
<td>The Payable Tagata is no longer in Issued state and cannot be modified.'</td>
</tr>
<tr>
<td>CANT_MODIFY_TAGATA</td>
<td>The Receivable Tagata is no longer in Holding state and cannot be modified.</td>
</tr>
<tr>
<td>CANT_MODIFY_TEGATA</td>
<td>The Payable Tagata is linked to bills and cannot be modified.'</td>
</tr>
<tr>
<td>CANT_MODIFY_VOID_TRANS</td>
<td>The G/L impact of a voided transaction cannot be changed.</td>
</tr>
<tr>
<td>CANT_MODIFY_VOID_TRANS</td>
<td>You may not change the GL impact on a voided {1: transaction type}.</td>
</tr>
<tr>
<td>CANT_MOVE_REALIZED_GAINLOSS</td>
<td>You cannot move a Realized Gain/Loss transaction to a date before either the source or the payment transaction.</td>
</tr>
<tr>
<td>CANT_PAY_TAGATA</td>
<td>Endorsed Tagata can only be paid on or after its maturity date.</td>
</tr>
<tr>
<td>CANT_PAY_TAGATA</td>
<td>Payable Tagata can only be paid on or after its maturity date.</td>
</tr>
<tr>
<td>CANT_PROCESS_IMG</td>
<td>Faceless PDF Library unable to process image. Image DPI:{1}.</td>
</tr>
<tr>
<td>CANT_RCEIV BEFORE_FULFILL</td>
<td>The item receipt for a transfer order line can not occur before the item fulfillment.</td>
</tr>
<tr>
<td>CANT_RCEIV_ITEM</td>
<td>A mark shipped fulfillment line has been processed against item {1:item name}. This item cannot be received.</td>
</tr>
<tr>
<td>CANT_RECEIVE_TAGATA</td>
<td>Receivable Tagata can only be collected on or after its maturity date.</td>
</tr>
<tr>
<td>CANT_REJECT_ORDER</td>
<td>You cannot reject this order because it has already been approved.</td>
</tr>
<tr>
<td>CANT_REMOV_ALL_FULFILMNT_LINKS</td>
<td>You may not modify this sales order in such a way that it removes all links to any fulfillment. The modifications you made would leave the fulfillment &lt;a href=&quot;app/accounting/transactions/transaction.nl?id={1}&quot;&gt;{2}&lt;/a&gt; unlinked.</td>
</tr>
<tr>
<td>CANT_REMOV_ITEM_SUB</td>
<td>You may not remove a subsidiary from an item that is a member of an assembly, group, or kit item if the parent item is available in that subsidiary.</td>
</tr>
<tr>
<td>CANT_REMOVE_ACH_PAY_METHOD</td>
<td>ACH payment methods cannot be removed</td>
</tr>
<tr>
<td>CANT_REMOVE_APPROVAL</td>
<td>The accounting approval cannot be removed from this expense report because some of its lines have already been invoiced to the customer.</td>
</tr>
<tr>
<td>CANT_REMOVE_DOMAIN</td>
<td>You are trying to remove a domain that is referenced by {1} CRM template(s). Please first clear the domain from CRM templates before trying to remove it.</td>
</tr>
<tr>
<td>CANT_REMOVE_NEXUS</td>
<td>A nexus cannot be removed from a subsidiary if the nexus is associated with a transaction.</td>
</tr>
<tr>
<td>CANT_REMOVE_SCHEDUL</td>
<td>You have attempted to remove an active schedule from a plan. Removing this participant is not permitted after commissions against the plan have been generated.</td>
</tr>
<tr>
<td>CANT_REMOVE_SUB</td>
<td>You cannot remove subsidiary: {1} because this record is used on a transaction for subsidiary: {1}.</td>
</tr>
<tr>
<td>CANT_REMOVE_SUB</td>
<td>You attempted to remove one or more subsidiaries from this item, but the item appears in at least one transaction in those subsidiaries. To remove a subsidiary from the item, make sure the item does not appear in any transactions for that subsidiary.</td>
</tr>
<tr>
<td>CANT_RESUBMIT_FAILED_DPLYMNT</td>
<td>You cannot submit a deployment for execution whose status is set to Scheduled.</td>
</tr>
<tr>
<td>CANT_RETURN_FLD</td>
<td>Can not return field {1}. Reason: {2}</td>
</tr>
<tr>
<td>CANT_RETURN_USED_GIFT_CERT</td>
<td>Used gift certificates can not be returned.</td>
</tr>
<tr>
<td>CANT_REV_REC_BODY_AND_LINE</td>
<td>The Revenue Recognition fields must be specified at EITHER the transaction body or the item line level, and may NOT be specified at both levels.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CANT_REVERSE_AUTH</td>
<td>Card type doesn't allow reversals. No resolution. Authorization cannot be reversed.</td>
</tr>
<tr>
<td>CANT_SCHEDUL_RECUR_EVENT</td>
<td>Because the number of days in each month differs, recurring monthly events cannot be scheduled after the 28th.</td>
</tr>
<tr>
<td>CANT_SEND_EMAIL</td>
<td>Unable to send notification email</td>
</tr>
<tr>
<td>CANT_SEND_EMAIL</td>
<td>Unable to send notification email to support rep</td>
</tr>
<tr>
<td>CANT_SET_CLOSE_DATE</td>
<td>Unable to set expected close date of prospect/lead based on current estimates/opportunities.</td>
</tr>
<tr>
<td>CANT_SET_INTERNALID</td>
<td>You cannot set internalId with upsert.</td>
</tr>
<tr>
<td>CANT_SET_STATUS</td>
<td>Unable to set status of prospect/lead based on current estimates.</td>
</tr>
<tr>
<td>CANT_SWITCH_ROLES_FROM_LOGIN</td>
<td>Role switching is not allowed from this login.</td>
</tr>
<tr>
<td>CANT_SWITCH_SHIP_METHOD</td>
<td>Switching the shipping method to another carrier is an unsupported operation, because it requires reloading the item fulfillment form for that carrier.</td>
</tr>
<tr>
<td>CANT_UPDATE_ACCTNG_PRDS</td>
<td>You cannot update accounting periods using SuiteScript or SOAP web services. Go to Setup &gt; Accounting &gt; Manage G/L &gt; Manage Accounting Periods.</td>
</tr>
<tr>
<td>CANT_UPDATE_AMT</td>
<td>The amount on lines containing partially/fully recognized schedules can not be changed</td>
</tr>
<tr>
<td>CANT_UPDATE_DYNAMIC_GROUP</td>
<td>You cannot update dynamic groups. Instead you must modify the saved search associated with the group</td>
</tr>
<tr>
<td>CANT_UPDATE_FLDR</td>
<td>These predefined folders cannot be updated</td>
</tr>
<tr>
<td>CANT_UPDATE_LINKED_TRANS_LINES</td>
<td>You cannot update linked transaction lines</td>
</tr>
<tr>
<td>CANT_UPDATE_PRODUCT_FEED</td>
<td>This item has multiple product feeds. SOAP web services schema version 2.6 or greater is required to modify product feeds for this item</td>
</tr>
<tr>
<td>CANT_UPDATE_RECRD_HAS_CHANGED</td>
<td>Cannot update bug. Record has changed since you last retrieved it.</td>
</tr>
<tr>
<td>CANT_UPDATE_RECUR_EVENT</td>
<td>Event &lt;id {1}&gt; contains recurrence patterns that are not supported in your client application. You are not allowed to update recurrence pattern on this event. Contact your software vendor for the latest Web Services upgrade.</td>
</tr>
<tr>
<td>CANT_UPDATE_ROOT_CATEGORY</td>
<td>Can not update root level website categories through SOAP web services.</td>
</tr>
<tr>
<td>CANT_UPDATE_STATUS_TYPE</td>
<td>You cannot update the only status of type {1}</td>
</tr>
<tr>
<td>CANT_VERIFY_CARD</td>
<td>Card Verify not supported. Retry request.</td>
</tr>
<tr>
<td>CANT_VOID_TRANS</td>
<td>You cannot void this transaction because it is linked to by one or more transactions such as payments. You must delete or void those transactions first</td>
</tr>
<tr>
<td>CARD_EXPIRED</td>
<td>Expired Card. Re-submit with valid expiration date.</td>
</tr>
<tr>
<td>CARD_ID_REQD</td>
<td>Card ID required. Provide a valid card ID.</td>
</tr>
<tr>
<td>CASE_ALREADY_ASSIGNED</td>
<td>This case cannot be grabbed because it is already assigned to another rep. To view the case, go back and click on the case number.</td>
</tr>
<tr>
<td>CASE_DSNT_EXIST</td>
<td>Case doesn't exist or no customer is associated with case.</td>
</tr>
<tr>
<td>CASE_NOT_GROUP_MEMBER</td>
<td>{1} this case record does not belong to your group.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CASH_SALE_EDIT_DISALLWD</td>
<td>This cash sale cannot be edited while it has an Automated Clearing House transmission in process.</td>
</tr>
<tr>
<td>CC_ACCT_REQD</td>
<td>You must have a credit card account to perform this operation.</td>
</tr>
<tr>
<td>CC_ACCT_REQD</td>
<td>You must have a credit card account to perform this operation. Click &lt;a href=&quot;/app/accounting/account/account.nl&quot;&gt;here&lt;/a&gt; to add one.</td>
</tr>
<tr>
<td>CC_ALREADY_SAVED</td>
<td>That credit card is already saved. Please use the saved credit card.</td>
</tr>
<tr>
<td>CC_EMAIL_ADDRESS_REQD</td>
<td>Please go back and provide an email address to CC store orders to.</td>
</tr>
<tr>
<td>CC_NUM_REQD</td>
<td>Please provide a credit card number.</td>
</tr>
<tr>
<td>CC_PROCESSOR_ERROR</td>
<td>An unexpected error occurred while processing the credit card through Merchant e-Solutions. (reason code = {1}). Please contact NetSuite support.</td>
</tr>
<tr>
<td>CC_PROCESSOR_NOT_FOUND</td>
<td>A suitable credit card processor was not found for this transaction.</td>
</tr>
<tr>
<td>CC_SECURITY_CODE_REQD</td>
<td>This transaction requires the Credit Card Security Code. Please enter the required value in the {1} field and re-submit.</td>
</tr>
<tr>
<td>CERT_AUTH_EXPD</td>
<td>CA expired on {1}</td>
</tr>
<tr>
<td>CERT_EXPD</td>
<td>Certificate expired on {1}</td>
</tr>
<tr>
<td>CERT_UNAVAILABLE</td>
<td>Certificate unavailable (most likely has not been presented by client)</td>
</tr>
<tr>
<td>CHANGE_PMT_DATE_AND_REAPPROVE</td>
<td>The payment is more than 30 days past due and has NOT been sent. Edit the payment to change the date and reapprove.</td>
</tr>
<tr>
<td>CHAR_ERROR</td>
<td>Character error on Line# (1) Column# (2) (Byte # (3)). (4)</td>
</tr>
<tr>
<td>CHECKOUT_EMAIL_REQD</td>
<td>Please go back and provide an email address to email checkout errors to.</td>
</tr>
<tr>
<td>CITY_REQD</td>
<td>City is a required field and it cannot be null or empty.</td>
</tr>
<tr>
<td>CLASS_ALREADY_EXISTS</td>
<td>A class already exists with that name. Go &lt;a href=&quot;javascript:history.go(-1);&quot;&gt;back&lt;/a&gt;, change the name and resubmit.</td>
</tr>
<tr>
<td>CLASS_OR_DEPT_OR_CUST_REQD</td>
<td>only one of class, cust, and dept can be non-null</td>
</tr>
<tr>
<td>CLEAR_AUTOCALC</td>
<td>For items that use the time phased replenishment method, you must clear the Auto-Calculate checkbox next to the Reorder Point and Preferred Stock Level fields. The mass update cannot be performed unless these settings are changed.</td>
</tr>
<tr>
<td>CLOSE_PREVIOUS_PERIOD</td>
<td>Please close previous period before working on this one.</td>
</tr>
<tr>
<td>CLOSED_TRAN_PRD</td>
<td>The G/L impact of a transaction in a closed period cannot be changed.</td>
</tr>
<tr>
<td>CLOSED_TRAN_PRD</td>
<td>You cannot move a transaction to or from a closed period.</td>
</tr>
<tr>
<td>COGS_ERROR</td>
<td>COGS lines not cleaned up</td>
</tr>
<tr>
<td>COGS_ERROR</td>
<td>Cost of Goods Sold lines not in balance</td>
</tr>
<tr>
<td>COGS_ERROR</td>
<td>LIFO/FIFO COGS count does not equal the number of items requested COGS ERROR 9765 itemsLinked={1}, itemsTotal={2} kdoc={3}, nid={4}</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>COMMSSN_ALREADY_CALCLTD</td>
<td>You have attempted to remove an active sales participant from a plan. Removing this participant is not permitted after commissions against the plan have been generated.</td>
</tr>
<tr>
<td>COMMSSN_FEATURE_DISABLED</td>
<td>You have not enabled the Commissions feature.</td>
</tr>
<tr>
<td>COMMSSN_FEATURE_DISABLED</td>
<td>You have not enabled the Partner Commissions/Royalties feature.</td>
</tr>
<tr>
<td>COMMSSN_PAYROLL_ITEM_REQD</td>
<td>A commission payroll item must be added for each employee to be processed through payroll.</td>
</tr>
<tr>
<td>COMP_DELETED_OR_MERGED</td>
<td>The company you try to attach the context to has been deleted or merged.</td>
</tr>
<tr>
<td>COMPANION_PROP_REQD</td>
<td>Error - Items do not have companion property (column) {1}</td>
</tr>
<tr>
<td>COMPANY_FLD_REQD</td>
<td>The Company field is required for COD shipments. Go to Setup &gt; Accounting &gt; Shipping, set the Company field on your FedEx Registration record and re-submit.</td>
</tr>
<tr>
<td>CONCUR_BILLPAY_JOB_DISALLWD</td>
<td>Your account currently has a bill pay approval job in progress. Only one bill pay approval job per account is allowed at a time. Please wait until this process completes before submitting another group of payments for approval. Visit the status page to track the progress of the current job.</td>
</tr>
<tr>
<td>CONCUR_BULK_JOB_DISALLWD</td>
<td>This Account is already running a bulk processing job. Please visit the status page to track the progress of the current job.</td>
</tr>
<tr>
<td>CONCUR_MASS_UPDATE_DISALLWD</td>
<td>A mass update is currently running in this account. Please try again in a few minutes.</td>
</tr>
<tr>
<td>CONCUR_SEARCH_DISALLWD</td>
<td>Search aborted by concurrent {1} search. Only one search may run at a time.</td>
</tr>
<tr>
<td>CONSLD_PRNT_AND_CHILD_DISALLWD</td>
<td>A company can be a consolidated child or a consolidated parent but not both.</td>
</tr>
<tr>
<td>CONTACT_ALREADY_EXISTS</td>
<td>A contact record with this name already exists. Every contact record must have a unique name.</td>
</tr>
<tr>
<td>CONTACT_ALREADY_EXISTS</td>
<td>A contact with the name [{1}] already exists.</td>
</tr>
<tr>
<td>CONTACT_NOT_GROUP_MEMBR</td>
<td>{1} this contact does not belong to your group.</td>
</tr>
<tr>
<td>COOKIES_DISABLED</td>
<td>You have disabled cookies from being stored on your computer or turned off per-session cookies. Please enable this feature and try again.</td>
</tr>
<tr>
<td>COUNTRY_STATE_MISMATCH</td>
<td>The country and state/province are mismatched, the country is {1} and the state/province is {2}. Please enter a state/province short name that matches the country.</td>
</tr>
<tr>
<td>CREATEDFROM_REQD</td>
<td>Please enter a value for createdFrom.</td>
</tr>
<tr>
<td>CREDITS_DISALLWD</td>
<td>Credits Not Allowed. Contact Merchant e-Solutions to have credits enabled.</td>
</tr>
<tr>
<td>CRNCY_MISMATCH_BASE_CRNCY</td>
<td>The currency you are registered to use is different from the base currency of this company.</td>
</tr>
<tr>
<td>CRNCY_NOT_UPDATED</td>
<td>The following currencies were not updated: {1}</td>
</tr>
<tr>
<td>CRNCY_RCRD_DELETED</td>
<td>This currency record has been deleted. You can create a new currency record at Lists &gt; Currencies.</td>
</tr>
<tr>
<td>CRNCY_REQD</td>
<td>currency expected for pricing element</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CSC_SETUP_REQD</td>
<td>To display the CSC field on the form, you must enable the &quot;Use Card Security Code for Credit Card Transactions&quot; preference, located on the Setup &gt; Accounting Preferences task.</td>
</tr>
<tr>
<td>CSTM_FIELD_KEY_REQD</td>
<td>The specified custom field key is missing.</td>
</tr>
<tr>
<td>CSTM_FIELD_VALUE_REQD</td>
<td>The specified custom field value is missing.</td>
</tr>
<tr>
<td>CSV_DELIMITER_ERROR</td>
<td>Values in the following CSV file(s) are not comma separated, and cannot be imported. Please reformat the file(s) and try again. For instructions, visit this help topic.</td>
</tr>
<tr>
<td>CUST_ARLEADY_HAS_ACCT</td>
<td>Attempting to provision a new account to a customer with an existing account.</td>
</tr>
<tr>
<td>CUST_CNTR_USER_ACCESS_ONLY</td>
<td>This form is only accessible to customer center users.</td>
</tr>
<tr>
<td>CUST_LEAD_NOT_GROUP_MEMBR</td>
<td>{1} This customer or lead does not belong to your group.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>An unexpected error occurred while processing the credit card through CyberSource (reason code = {1}). Please contact NetSuite Customer Support.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The credit card was declined by the card issuer. Use a different card for payment or contact the card issuer for more information.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The credit card transaction was denied by the issuing bank. Please try another card or contact the card issuer for more information.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The credit card has expired or the expiration date does not match the date on file with the card issuer. Please correct the expiration date or try another card.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The credit card transaction was denied due to insufficient funds. Please try another card or contact the card issuer for more information.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The credit card transaction could not be completed because the issuing bank was not available. Please try another card or wait a few minutes and try again.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Inactive card or card not authorized for card-not-present transactions. Please try another card or contact the card issuer for more information.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The card has reached the credit limit. Please try another card or contact the card issuer for more information.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Invalid card verification number. Please check to make sure you have provided the correct card verification number.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Invalid credit card account number. Please check to make sure you have provided the correct credit card account number.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The type of credit card provided is not accepted by this merchant. Please try another card or contact the merchant for more information.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Successful transaction.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The request is missing one or more required fields. Possible action: See the reply fields missingField_0...N for which fields are missing. Resend the request with the complete information.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>One or more fields in the request contains invalid data. Possible action: See the reply fields invalidField_0...N for which fields are invalid. Resend the request with the correct information.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The merchantReferenceCode sent with this authorization request matches the merchantReferenceCode of another authorization request that you sent in the last 15 minutes. Possible action: Resend the request with a unique merchantReferenceCode value.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Error: General system failure. See the documentation for your CyberSource client (SDK) for information about how to handle retries in the case of system errors.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Error: The request was received but there was a server timeout. This error does not include timeouts between the client and the server. Possible action: To avoid duplicating the order, do not resend the request until you have reviewed the order status in the Business Center. See the documentation for your CyberSource client (SDK) for information about how to handle retries in the case of system errors.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Error: The request was received, but a service did not finish running in time. Possible action: To avoid duplicating the order, do not resend the request until you have reviewed the order status in the Business Center. See the documentation for your CyberSource client (SDK) for information about how to handle retries in the case of system errors.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The issuing bank has questions about the request. You do not receive an authorization code programmatically, but you might receive one verbally by calling the processor. Possible action: Call your processor or the issuing bank to possibly receive a verbal authorization. For contact phone numbers, refer to your merchant bank information.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Expired card. You might also receive this if the expiration date you provided does not match the date the issuing bank has on file. Possible action: Request a different card or other form of payment.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>General decline of the card. No other information provided by the issuing bank. Possible action: Request a different card or other form of payment.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Insufficient funds in the account. Possible action: Request a different card or other form of payment.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Stolen or lost card. Possible action: Review the customers information and determine if you want to request a different card from the customer.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Issuing bank unavailable. Possible action: Wait a few minutes and resend the request.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Inactive card or card not authorized for card-not-present transactions. Possible action: Request a different card or other form of payment.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The card has reached the credit limit. Possible action: Request a different card or other form of payment.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Invalid card verification number. Possible action: Request a different card or other form of payment.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The customer matched an entry on the processors negative file. Possible action: Review the order and contact the payment processor.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Invalid account number. Possible action: Request a different card or other form of payment.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The card type is not accepted by the payment processor. Possible action: Request a different card or other form of payment. Also, check with CyberSource Customer Support to make sure your account is configured correctly.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>General decline by the processor. Possible action: Request a different card or other form of payment.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>There is a problem with your CyberSource merchant configuration. Possible action: Do not resend the request. Contact Customer Support to correct the configuration problem.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The requested amount exceeds the originally authorized amount. Occurs, for example, if you try to capture an amount larger than the original authorization amount. This reason code only applies if you are processing a capture through the API. See Using the API for Captures and Credits. Possible action: Issue a new authorization and capture request for the new amount.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Processor failure. Possible action: Tell the customer the payment processing system is unavailable temporarily, and to try their order again in a few minutes.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The authorization has already been captured. This reason code only applies if you are processing a capture through the API. See Using the API for Captures and Credits. Possible action: No action required.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The requested transaction amount must match the previous transaction amount. This reason code only applies if you are processing a capture or credit through the API. See Using the API for Captures and Credits. Possible action: Correct the amount and resend the request.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The card type sent is invalid or does not correlate with the credit card number. Possible action: Ask your customer to confirm that the card is the type that they indicated in your Web store, then resend the request.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The request ID is invalid. This reason code only applies when you are processing a capture or credit through the API. See Using the API for Captures and Credits. Possible action: Request a new authorization, and if successful, proceed with the capture.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>You requested a capture through the API, but there is no corresponding, unused authorization record. Occurs if there was not a previously successful authorization request or if the previously successful authorization has already been used by another capture request. This reason code only applies when you are processing a capture through the API. See Using the API for Captures and Credits. Possible action: Request a new authorization, and if successful, proceed with the capture.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The capture or credit is not voidable because the capture or credit information has already been submitted to your processor. Or, you requested a void for a type of transaction that cannot be voided. This reason code applies only if you are processing a void through the API. See Using the API for Voids for information about voids. Possible action: No action required.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>You requested a credit for a capture that was previously voided. This reason code applies only if you are processing a void through the API. See Using the API for Voids for information about voids. Possible action: No action required.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Error: The request was received, but there was a timeout at the payment processor. Possible action: To avoid duplicating the transaction, do not resend the request until you have reviewed the transaction status in the Business Center.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>The authorization request was approved by the issuing bank but declined by CyberSource based on your Smart Authorization settings. Possible action: Do not capture the authorization without further review. Review the ccAuthReply_avsCode, ccAuthReply_cvCode, and ccAuthReply_authFactorCode fields to determine why CyberSource rejected the request.</td>
</tr>
<tr>
<td>CYBERSOURCE_ERROR</td>
<td>Unable to process credit card transaction. The code returned from CyberSource (1) is not a recognized reason code. Please contact NetSuite support.</td>
</tr>
<tr>
<td>CYCLE_IN_PROJECT_PLAN</td>
<td>The changes made to this entity have cause a cycle in the project plan. Select a different parent and/or predecessors to avoid the cycle.</td>
</tr>
<tr>
<td>DASHBOARD_LOCKED</td>
<td>Your dashboard has been set up and locked by an administrator. Please contact them for details.</td>
</tr>
<tr>
<td>DATA_MUST_BE_UNIQUE</td>
<td>The update failed because every entry in this column must be unique.</td>
</tr>
<tr>
<td>DATA_REQD</td>
<td>You need to provide a proper value for the required field: (1).</td>
</tr>
<tr>
<td>DATA_REQD</td>
<td>You are missing the following required field(s):(1)                                                                ----------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DATEEXPECTED</td>
<td>You entered ('1)' into a field where a calendar date was expected. Please go back and change this value to the correct date.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DATE_PARAM_REQD</td>
<td>missing date parameter</td>
</tr>
<tr>
<td>DATE_PRD_MISMATCH</td>
<td>Your transaction date does not fall between the start and end dates of your accounting period.</td>
</tr>
<tr>
<td>DEFAULT_CUR_REQD</td>
<td>Default currency cannot be null</td>
</tr>
<tr>
<td>DEFAULT_EXPENSE_ACCT_REQD</td>
<td>A default expense account must be specified to activate items on the list. Go to Setup &gt; Set Up Payroll and click the Default Accounts subtab. In the Payroll Expenses Account field, choose a default general ledger account for your payroll expenses. Then, click Save.</td>
</tr>
<tr>
<td>DEFAULT_ISSUE_OWNER_REQD</td>
<td>There is no default owner for the issue role {1}. This operation cannot be completed until this is corrected.</td>
</tr>
<tr>
<td>DEFAULT_LIAB_ACCT_REQD</td>
<td>A default liability account must be specified to activate items on the list. Go to Setup &gt; Set Up Payroll and click the Default Accounts subtab. In the Payroll Liabilities Account field, choose a default general ledger account for your payroll liabilities. Then, click Save.</td>
</tr>
<tr>
<td>DEFAULT_ROLE_REQD</td>
<td>Login Failed because you do not have a default role for the company and email entered. Please Try Again.</td>
</tr>
<tr>
<td>DEFAULT_TYPE_DELETE_DISALLWD</td>
<td>You cannot delete default types</td>
</tr>
<tr>
<td>DEFERRAL_ACCT_REQD</td>
<td>Lines with amortization templates must have a deferral account.</td>
</tr>
<tr>
<td>DEFERRAL_ACCT_REQD</td>
<td>Lines with revenue recognition templates must have a deferral account.</td>
</tr>
<tr>
<td>DEFERRED_REV_REC_ACCT_REQD</td>
<td>The {1} item does not have a Deferred Revenue Account specified. Please assign the item a Deferred Revenue Account using the standard User Interface, and then re-import the transaction.</td>
</tr>
<tr>
<td>DEPT_IN_USE</td>
<td>Your classes cannot be converted to departments because your existing department records are referred to by transactions or other records. These department records cannot be overwritten.</td>
</tr>
<tr>
<td>DFRNT_SWAP_PRICE_LEVELS_REQD</td>
<td>Please select different price levels to swap prices.</td>
</tr>
<tr>
<td>DISALLWD_IP_ADDRESS</td>
<td>The specified IP address rules must allow the login of your current IP Address. Your current IP address is {1}. For information on entering IP address rules, click Help at the top of the page.</td>
</tr>
<tr>
<td>DISCOUNT_ACCT_SETUP_REQD</td>
<td>Please &lt;a href='/app/setup/acctsetup.nl'&gt;Set Up Discount Accounts&lt;/a&gt; first.</td>
</tr>
<tr>
<td>DISCOUNT_DISALLWD</td>
<td>You have attempted to save this transaction with one or more discounts and where all items have Permit Discount = Never. You must change one of the items to permit a discount, add a new item without the restriction or remove the discount from the transaction.</td>
</tr>
<tr>
<td>DISCOUNT_DISALLWD_VSOE</td>
<td>Posting discounts are not allowed on items in VSOE bundles.</td>
</tr>
<tr>
<td>DISCOUNT_EXCEED_TOTAL</td>
<td>Discount can not exceed item total.</td>
</tr>
<tr>
<td>DISTRIBUT_REQD_ONE_DAY_BFORE</td>
<td>All items must be distributed at least one day before they may be transferred.</td>
</tr>
<tr>
<td>DOMAIN_IN_USE</td>
<td>The domain {1} is already in use</td>
</tr>
<tr>
<td>DOMAIN_WEBSITE_REQD</td>
<td>Please select a Web Site for domain {1}</td>
</tr>
<tr>
<td>DROP_SHIP_ERROR</td>
<td>The following error occurred when updating the quantity on the drop ship {1:transaction type}: &lt;p&gt;{2:error message}&lt;/p&gt;</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DROP_SHIP_ERROR</td>
<td>The transaction was successfully saved, but an error occurred while running user events and email alerts after updating the drop ship (1:transaction type)(s)</td>
</tr>
<tr>
<td>DROP_SHIP_ERROR</td>
<td>The transaction was successfully saved, but an error occurred while running user events after updating the drop ship (1:transaction type)(s)</td>
</tr>
<tr>
<td>DROP_SHIP_OR_SPECIAL_ORD_ALLWD</td>
<td>Items can be Drop Ship or Special Order but not both</td>
</tr>
<tr>
<td>DUE_DATE_BFORE_START_DATE</td>
<td>Due date occurs before start date</td>
</tr>
<tr>
<td>DUE_DATE_REQD</td>
<td>Please enter a value for {1} Due Date</td>
</tr>
<tr>
<td>DUP_ACCT_NAME</td>
<td>The account name you have chosen is already used.&lt;br&gt;Go &lt;a href='javascript:history.go(-1);';&gt;back&lt;/a&gt;, change the name and resubmit.</td>
</tr>
<tr>
<td>DUP_ACCT_NOT_ALLWD</td>
<td>You may not use duplicate accounts on an item.</td>
</tr>
<tr>
<td>DUP_ACCT_NUM</td>
<td>The account number you have chosen is already used.</td>
</tr>
<tr>
<td>DUP_ACCT_NUM</td>
<td>The account number you have chosen is already used.&lt;br&gt;Go &lt;a href='javascript:history.go(-1);';&gt;back&lt;/a&gt;, change the number and resubmit.</td>
</tr>
<tr>
<td>DUP_ACCT_ON_TRANS</td>
<td>This transaction has duplicate accounts. The main line of the transaction and the line labeled '{1}' both use the account named '{2}'.</td>
</tr>
<tr>
<td>DUP_BIN</td>
<td>A bin already exists with that name. Go back, change the name, and resubmit.</td>
</tr>
<tr>
<td>DUP_BIN</td>
<td>There is already another bin with that number. Please choose a bin number that is not used by another bin.</td>
</tr>
<tr>
<td>DUP_BUNDLE_IN_ACCT</td>
<td>That bundle has already been copied or installed in this account.</td>
</tr>
<tr>
<td>DUP_BUNDLE_IN_ACCT</td>
<td>You cannot install this bundle as it is a copy of bundle {1} that you previously installed.</td>
</tr>
<tr>
<td>DUP_CATEGORY</td>
<td>This category already exists</td>
</tr>
<tr>
<td>DUP_CATEGORY_NAME</td>
<td>A category already exists with that name. Go &lt;a href='javascript:history.go(-1);';&gt;back&lt;/a&gt;, change the name and resubmit.</td>
</tr>
<tr>
<td>DUP_COLOR_THEME</td>
<td>This color theme already exists</td>
</tr>
<tr>
<td>DUP_CSTM_FIELD</td>
<td>This custom field already exists</td>
</tr>
<tr>
<td>DUP_CSTM_LAYOUT</td>
<td>This custom layout already exists</td>
</tr>
<tr>
<td>DUP_CSTM_LIST</td>
<td>There is already a Custom List or Custom List element with that name</td>
</tr>
<tr>
<td>DUP_CSTM_RCRD</td>
<td>There is already a Custom Record with that name</td>
</tr>
<tr>
<td>DUP_CSTM_RCRD_ENTRY</td>
<td>There is already a Custom Record Entry with that name</td>
</tr>
<tr>
<td>DUP_CSTM_TAB</td>
<td>This custom tab already exists</td>
</tr>
<tr>
<td>DUP_EMPL_EMAIL</td>
<td>There is already an employee with external access to this account using that email address. All employees with external access must have a unique email address for login purposes. Go &lt;a href='javascript:history.go(-1);';&gt;back&lt;/a&gt;, change the email address and resubmit.</td>
</tr>
<tr>
<td>DUP_EMPL_ENTITY_NAME</td>
<td>There is already an employee with external access to this account using that entity name. All employees with external access must have a unique entity name for login purposes. Go &lt;a href='javascript:history.go(-1);';&gt;back&lt;/a&gt;, change the entity name and resubmit.</td>
</tr>
<tr>
<td>DUP_EMPL_TMPLT</td>
<td>There is already an employee template with that name. Go &lt;a href='javascript:history.go(-1);';&gt;back&lt;/a&gt;, change the template name and resubmit.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DUP_ENTITY</td>
<td>This entity already exists.</td>
</tr>
<tr>
<td>DUP_ENTITY_EMAIL</td>
<td>There is already an external entity (eg. customer, vendor, or employee) with access to this account using that email address. All external entities with access must have a unique email address for login purposes.</td>
</tr>
<tr>
<td>DUP_ENTITY_NAME</td>
<td>There is already an external entity (eg. customer, vendor, or employee) with access to this account using that entity name. All external entities with access must have a unique entity name for login purposes.</td>
</tr>
<tr>
<td>DUP_FEDEX_ACCT_NUM</td>
<td>There is an existing NetSuite registration for FedEx account number {1}.</td>
</tr>
<tr>
<td>DUP_FINANCL_STATMNT_LAYOUT</td>
<td>This financial statement layout already exists.</td>
</tr>
<tr>
<td>DUP_INFO_ITEM</td>
<td>This information item already exists</td>
</tr>
<tr>
<td>DUP_ISSUE_NAME_OR_NUM</td>
<td>You cannot set {1:issue record name} {2:issue number} to be a duplicate of itself or one of its duplicates.</td>
</tr>
<tr>
<td>DUP_ITEM</td>
<td>Uniqueness error - there is already an item with that name or name/parent combination.</td>
</tr>
<tr>
<td>DUP_ITEM_LAYOUT</td>
<td>This item/category layout already exists</td>
</tr>
<tr>
<td>DUP_ITEM_NAME</td>
<td>There is already an item with that name.&lt;br&gt;Go &lt;a href=&quot;javascript:history.go(-1);&quot;&gt;back&lt;/a&gt;, change the name and resubmit.</td>
</tr>
<tr>
<td>DUP_ITEM_OPTION</td>
<td>A child item child with that combination of options already exists</td>
</tr>
<tr>
<td>DUP_TMPLT</td>
<td>This item/category template already exists</td>
</tr>
<tr>
<td>DUP_MATRIX_OPTN_ABBRV</td>
<td>Matrix option '{1}' already uses that abbreviation. Please choose another.</td>
</tr>
<tr>
<td>DUP_MEMRZD_TRANS</td>
<td>There is already a Memorized Transaction with that name.&lt;br&gt;Go &lt;a href=&quot;javascript:history.go(-1);&quot;&gt;back&lt;/a&gt;, change the name and resubmit.</td>
</tr>
<tr>
<td>DUP_NAME</td>
<td>That name is already in use.&lt;br&gt;Go &lt;a href=&quot;javascript:history.go(-1);&quot;&gt;back&lt;/a&gt;, change the name and resubmit.</td>
</tr>
<tr>
<td>DUP_PAYROLL_ITEM</td>
<td>There is already a payroll item named {1}</td>
</tr>
<tr>
<td>DUP_PRESNTN_CAT</td>
<td>This presentation category already exists</td>
</tr>
<tr>
<td>DUP_RCRD</td>
<td>A {1} already exists with that name. Go &lt;a href=&quot;javascript:history.go(-1);&quot;&gt;back&lt;/a&gt;, change the name and resubmit.</td>
</tr>
<tr>
<td>DUP_RCRD</td>
<td>Matched more than one record for {1}</td>
</tr>
<tr>
<td>DUP_RCRD</td>
<td>Matched more than one record (internalIds {1} and {2})</td>
</tr>
<tr>
<td>DUP_RCRD</td>
<td>This record already exists</td>
</tr>
<tr>
<td>DUP_RCRD_LINK</td>
<td>Link to that record already exists</td>
</tr>
<tr>
<td>DUP_SALES_TAX_ITEM</td>
<td>You have entered a duplicate Sales Tax Item.&lt;br&gt;Go &lt;a href=&quot;javascript:history.go(-1);&quot;&gt;back&lt;/a&gt;, change the name, city, state or zip code and resubmit.</td>
</tr>
<tr>
<td>DUP_SHIPPING_ITEM</td>
<td>You have entered a duplicate Shipping Item.&lt;br&gt;Go &lt;a href=&quot;javascript:history.go(-1);&quot;&gt;back&lt;/a&gt;, change the name and resubmit.</td>
</tr>
<tr>
<td>DUP_SHORT_NAME</td>
<td>Duplicate short name</td>
</tr>
<tr>
<td>DUP_SITE_THEME</td>
<td>This site theme already exists</td>
</tr>
<tr>
<td>DUP_SOURCE_ACCT</td>
<td>Duplicate source accounts are not allowed.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>DUP_TAX_CODE</td>
<td>You have entered a duplicate Tax Code. Go back, change the name and resubmit.</td>
</tr>
<tr>
<td>DUP_TAX_CODE</td>
<td>You have entered a duplicate Tax Code. Go back, change the name, city, state or zip code and resubmit.</td>
</tr>
<tr>
<td>DUP_TRACKING_NUM</td>
<td>You entered the following tracking number twice: {1}. Note that a single tracking number may not contain spaces or commas. A space or comma will be interpreted as the separator between different tracking numbers. For example, &quot;1029 3847 465&quot; will be interpreted as 3 different tracking numbers. It should be entered without spaces: &quot;10293847465&quot;.</td>
</tr>
<tr>
<td>DUP_TRANS</td>
<td>Duplicate Trans. Unable to locate, no match.</td>
</tr>
<tr>
<td>DUP_UPS_ACCT_NUM</td>
<td>There is an existing NetSuite registration for UPS account number {1}.</td>
</tr>
<tr>
<td>DUP_VENDOR_EMAIL</td>
<td>There is already a vendor with external access to this account using that email address. All vendors with external access must have a unique email address for login purposes. Go back, change the email address and resubmit.</td>
</tr>
<tr>
<td>DUP_VENDOR_NAME</td>
<td>There is already a vendor using that entity name. All vendors must have a unique entity name. Go back, change the entity name and resubmit.</td>
</tr>
<tr>
<td>DUPLICATE_INVENTORY_NUM</td>
<td>Duplicate inventory number found in entry: {1}</td>
</tr>
<tr>
<td>DUPLICATE_INVENTORY_NUM</td>
<td>Duplicate inventory number found on different lines of transaction</td>
</tr>
<tr>
<td>DUPLICATE_KEYS</td>
<td>This record contains duplicated key or keys. Please correct it before next update.</td>
</tr>
<tr>
<td>DUPLICATE_NAME_FOR_PRD</td>
<td>Please choose a different period name. &quot;{1}&quot; is already taken.</td>
</tr>
<tr>
<td>DUPLICATE_NAME_FOR_ROLE</td>
<td>Please choose a different role name. &quot;{1}&quot; is already taken.</td>
</tr>
<tr>
<td>EARNING_ITEM_REQD</td>
<td>At least one employee in this payroll has no earning items. Please make sure that every employee has at least one earning item.</td>
</tr>
<tr>
<td>EDITION_DSNT_SUPRT_WORLDPAY</td>
<td>WorldPay is not supported in this edition.</td>
</tr>
<tr>
<td>EIN_OR_TIN_REQD</td>
<td>You must set either the Employer Identification Number (EIN) or SSN/TIN (Social Security Number, Tax ID Number) to complete this fulfillment. Please go to Setup &gt; Company Information to set either of these fields and re-submit.</td>
</tr>
<tr>
<td>EMAIL_ADDRS_REQD</td>
<td>Please enter your email address</td>
</tr>
<tr>
<td>EMAIL_ADDRS_REQD_TO_NOTIFY</td>
<td>Please enter an email address for this company. A notification email will be sent when this case record is saved.</td>
</tr>
<tr>
<td>EMAIL_ADDRS_REQD_TO_NOTIFY</td>
<td>The recipient you are sending this email to does not have an email address. Please enter one and try again</td>
</tr>
<tr>
<td>EMAIL_REQ_HANDLER_ERROR</td>
<td>An error occurred while instantiating an email requesthandler</td>
</tr>
<tr>
<td>EMAIL_REQ_HANDLER_ERROR</td>
<td>An Error occurred while performing system-level validation of email request for &lt;{1}&gt; from &lt;{2}&gt;</td>
</tr>
<tr>
<td>EMAIL_REQ_HANDLER_ERROR</td>
<td>An Error occurred while POSTing data into requestHandler: {1}</td>
</tr>
<tr>
<td>EMAIL_REQ_HANDLER_ERROR</td>
<td>An error occurred while servicing an email requesthandler in state: {1}</td>
</tr>
<tr>
<td>EMAIL_REQ</td>
<td>You must enter a valid email address to email the transaction.</td>
</tr>
<tr>
<td>EMAIL_REQD_ACCT_PROVISION</td>
<td>Cannot provision an account without an email address for this customer: Was external access granted?</td>
</tr>
<tr>
<td>EMPL_IN_USE</td>
<td>You can't delete this employee, as commissions have been calculated for this employee.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EMPL_IN_USE</td>
<td>You can't delete this employee, as it is or has been referenced by other employees as a supervisor.</td>
</tr>
<tr>
<td>ERROR_DELETE_CARD_DATA</td>
<td>Failed to delete card data. Retry request.</td>
</tr>
<tr>
<td>ERROR_IN_TERRITORY_ASSGNMNT</td>
<td>Error Performing Initial Round_Robin Assignment for Territory: {1}</td>
</tr>
<tr>
<td>ERROR_IN_TERRITORY_ASSGNMNT</td>
<td>Error Performing Round_Robin Assignment for Territory: {1}</td>
</tr>
<tr>
<td>ERROR_PRCSSNG_TRANS</td>
<td>There were errors processing the selected transactions. Please process them individually for more information.</td>
</tr>
<tr>
<td>ERROR_REFUND_TRANS</td>
<td>Failed to refund International transaction. Retry request.</td>
</tr>
<tr>
<td>ERROR.Reverse_Auth</td>
<td>Failed to reverse International authorization. Retry request.</td>
</tr>
<tr>
<td>ERROR_SENDING_TRAN_EMAIL</td>
<td>The transaction was entered successfully, but an unexpected error occurred while sending the transaction email {1}</td>
</tr>
<tr>
<td>ERROR_VOID_TRANS</td>
<td>Failed to void International transaction. Retry request.</td>
</tr>
<tr>
<td>EVENT_ID_NOT_FOUND</td>
<td>Event ID not found</td>
</tr>
<tr>
<td>EXCEEDED_MAX_ALLWD_LOC</td>
<td>You have reached the maximum allowance of {1} location records. If you need to create additional location records, please contact our NetSuite Customer Support team for assistance</td>
</tr>
<tr>
<td>EXCEEDED_MAX_CONCUR_RQST</td>
<td>The maximum number of concurrent requests has been exceeded. Please try your request again when an existing session has completed.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_EMAILS</td>
<td>This account has {1} more bulk emails that can be sent. If you would like to purchase an additional block of emails, please contact your account manager.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_EMAILS</td>
<td>The merge exceeds the number of bulk merge emails allotted to your account this year. This account has {1} more bulk emails that can be sent this year. Please contact your NetSuite account manager to purchase additional block of emails.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_EMAILS</td>
<td>This campaign email event exceeds the number of emails ({1}) that can be sent per event without setting up a default campaign domain or specifying one on the campaign email template.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_EMAILS</td>
<td>This merge operation exceeds the number of emails ({1}) that can be sent per execution without setting up a bulk merge domain or specifying one on the email template.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_EMAILS</td>
<td>You cannot schedule more than {1} emails per year. If you'd like to purchase an additional block of emails, please contact your account manager.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_FEATURED_ITEMS</td>
<td>There is a limit of {1} featured items on this page.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_FIELD_LENGTH</td>
<td>Address line 1 cannot exceed 35 characters. Please check the shipper and recipient address to ensure the &quot;Address 1&quot; field is a maximum of 35 characters.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_FIELD_LENGTH</td>
<td>Address line 2 cannot exceed 35 characters. Please check the shipper and recipient address to ensure the &quot;Address 2&quot; field is a maximum of 35 characters.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_FIELD_LENGTH</td>
<td>The field {1} contained more than the maximum number ({2} ) of characters allowed.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_FIELD_LENGTH</td>
<td>The string &quot;{1}&quot; contained more than the maximum number of characters allowed.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_FIELD_LENGTH</td>
<td>Too many characters for a field</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EXCEEDED_MAX_MATRIX_OPTSNS</td>
<td>The total combination of subitems you have selected exceeds the maximum allowed of 2000. Please choose fewer options on the matrix tab.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_MATRIX_OPTSNS</td>
<td>The total combination of subitems you have selected exceeds the maximum allowed of 2000. Please choose fewer options on the matrix tab.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_PDF_ELEMENTS</td>
<td>There is a maximum of 100 custom elements allowed on a PDF layout.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_PDF_EXPORT_COL</td>
<td>PDF Export is limited to 30 columns.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_PIN_RETRIES</td>
<td>PIN entered incorrectly too often.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_RCRD</td>
<td>You have reached the maximum allowance of (1) (2) records. If you need to create additional (2) records, please contact our NetSuite Customer Support team for assistance</td>
</tr>
<tr>
<td>EXCEEDED_MAX_REPORT_COL</td>
<td>The option you selected in the Column field results in a report that exceeds the maximum number of columns allowed. Please select additional filters or select a shorter date/period range.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_REPORT_ROWS</td>
<td>Reports are limited to (1) rows. Please narrow your results.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_REPORT_SIZE</td>
<td>The results of this report are too large. Please narrow your results.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_SESSIONS</td>
<td>Maximum active sessions exceeded. Please wait 5 minutes and login again.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_SHIP_PACKAGE</td>
<td>The maximum number of custom shipping packages has been exceeded: (1). Please reduce item quantities to generate fewer packages, or enter the packages manually.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_TIME</td>
<td>The operation has exceeded maximum allowed time for completion. Operation aborted.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_TRANS_LINES</td>
<td>Transactions may not contain more than (1) lines.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_USERS_ALLWD</td>
<td>The changes you have made to this employee's access have caused you to exceed either your full access or Employee Center allowance. To make these changes you must either adjust the number of employees assigned the Employee Center role or contact your account representative to purchase additional licenses.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_USERS_ALLWD</td>
<td>Assigning this role would exceed your full access licenses (1)). To assign this role, you must remove another employee's full access roles or contact your account representative to purchase additional licenses.</td>
</tr>
<tr>
<td>EXCEEDED_MAX_USERS_ALLWD</td>
<td>The changes you have made to this employee's access have caused you to exceed either your full access or Retail User allowance. To make these changes you must either adjust the number of employees assigned the Retail Clerk role or contact your account representative to purchase additional licenses.</td>
</tr>
<tr>
<td>EXCEEDED_PER_TRANS_MAX</td>
<td>Exceeded per transaction maximum on account (1)</td>
</tr>
<tr>
<td>EXCEEDED_REQST_SIZE_LIMIT</td>
<td>You have exceeded the permitted request size limit (1))</td>
</tr>
<tr>
<td>EXCEEDS_ALLWD_LICENSES</td>
<td>Adding access for this user exceeds the number of licenses you have purchased. To add another user, you must first remove access from an existing user or contact NetSuite to purchase additional licenses.</td>
</tr>
<tr>
<td>EXCEEDS_ALLWD_LICENSES</td>
<td>Adding a (1) would exceed the number of licenses you have purchased. Please contact NetSuite for additional licenses.</td>
</tr>
<tr>
<td>EXPENSE_ENTRY_DISALLWD</td>
<td>(1) does not allow expense entry.</td>
</tr>
<tr>
<td>EXPIRED_SEARCH_CRITERIA</td>
<td>Your search criteria expired. The criteria for a specific search generally expire after 15 minutes of inactivity. Please return to the search definition page and re-submit your search.</td>
</tr>
<tr>
<td>EXT_CAT_LINK_SETUP_REQD</td>
<td>Error - you have not properly set up links from your External Catalog Site back into (1)!</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EXTERNALID_NOT_SUPPORTED</td>
<td>Field (1) does not support externalId</td>
</tr>
<tr>
<td>EXTERNALID_REQD</td>
<td>This operation requires a value for externalId.</td>
</tr>
<tr>
<td>FAILED_FEDEX_LABEL_VOID</td>
<td>Failed FedEx Label Void</td>
</tr>
<tr>
<td>FAILED_FORM_VALIDATION</td>
<td>Form validation failed. You cannot submit this record.</td>
</tr>
<tr>
<td>FAILED_UPS_LABEL VOID</td>
<td>Failed UPS Label Void</td>
</tr>
<tr>
<td>FAX_NUM_REQD</td>
<td>You must enter a fax number.</td>
</tr>
<tr>
<td>FAX_NUM_REQD</td>
<td>You must enter a fax number for this recipient before performing a fax merge operation.</td>
</tr>
<tr>
<td>FAX_NUM_REQD</td>
<td>You must enter a valid fax number to fax the transaction.</td>
</tr>
<tr>
<td>FAX_SETUP_REQD</td>
<td>Before you can send faxes, you need to go to the &lt;a href='/app/setup/printing.nl'&gt;Set Up Printing, Fax &amp; Email&lt;/a&gt; page and set up the fax service.</td>
</tr>
<tr>
<td>FEATURE_DISABLED</td>
<td>You do not have the correct features enabled to search on (1).</td>
</tr>
<tr>
<td>FEATURE_DISABLED</td>
<td>The ''(1)'' feature is not enabled in your (2) account.</td>
</tr>
<tr>
<td>FEATURE_DISABLED</td>
<td>The feature '1' required to access this page is not enabled in this account.</td>
</tr>
<tr>
<td>FEATURE_UNAVAILABLE</td>
<td>&lt;b&gt;{1} Trial does not allow access to this feature.&lt;/b&gt; If you would like more information about this feature, please contact your account manager.</td>
</tr>
<tr>
<td>FEATURE_UNAVAILABLE</td>
<td>Error - This business does not have the External Catalog Site feature enabled.</td>
</tr>
<tr>
<td>FEATURE_UNAVAILABLE</td>
<td>Test Drive does not allow access to this feature. If you would like more information about this feature, please contact your account manager.</td>
</tr>
<tr>
<td>FEATURE_UNAVAILABLE</td>
<td>That feature is only available to Plus users</td>
</tr>
<tr>
<td>FEATURE_UNAVAILABLE</td>
<td>The {1} feature is not available to your company.</td>
</tr>
<tr>
<td>FEATURE_UNAVAILABLE</td>
<td>This feature is not available to your company.</td>
</tr>
<tr>
<td>FED_ID_REQD</td>
<td>Must have Federal Identification Number to process 1099-MISC forms.</td>
</tr>
<tr>
<td>FED_WITHHOLDING_REQD</td>
<td>Your employee record does not have current Federal Withholding information.&lt;/p&gt;Please contact your supervisor to set up your record with the appropriate information.&lt;/p&gt;</td>
</tr>
<tr>
<td>FEDEX_ACCT_REQD</td>
<td>The FedEx Account Number has not been set.</td>
</tr>
<tr>
<td>FEDEX_CANT_INTEGRATE_FULFILL</td>
<td>The fulfillment cannot be integrated with {1} because the Shipping Integration Carrier is set to UPS.</td>
</tr>
<tr>
<td>FEDEX_DROPOFF_TYP_REQD</td>
<td>The FedEx Dropoff Type has not been set.</td>
</tr>
<tr>
<td>FEDEX_INVALID_ACCT_NUM</td>
<td>This account number was not recognized by FedEx. Please re-enter your account number, or contact FedEx to open a new account.</td>
</tr>
<tr>
<td>FEDEX_ITEM_CONTENTS_REQD</td>
<td>For international shipments, (1) requires specific information about the item contents.</td>
</tr>
<tr>
<td>FEDEX_METER_NOT_RETRIEVED</td>
<td>A FedEx Meter Number was not retrieved for account number {1}. Please try your request again in a few minutes.</td>
</tr>
<tr>
<td>FEDEX_METER_REQD</td>
<td>The FedEx Meter Number has not been set.</td>
</tr>
<tr>
<td>FEDEX_ONE_PACKG_ALLWD</td>
<td>The selected FedEx service allows only one package per fulfillment. If more than one package is required, please break up the shipment into multiple fulfillments of one package each.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>FEDEX_ORIGIN_COUNTRY_US_REQD</td>
<td>The origin country must be United States (US) for all Item Fulfillments when using a FedEx shipping method.</td>
</tr>
<tr>
<td>FEDEX_RATING_SRVC_UNAVAILBL</td>
<td>The FedEx rating services application is currently unavailable. Please try your request again in a few minutes.</td>
</tr>
<tr>
<td>FEDEX_REG_NOT_FOUND</td>
<td>A valid FedEx Registration was not found for the specified location:</td>
</tr>
<tr>
<td>FEDEX_SHIP_SRVC_REQD</td>
<td>The FedEx Shipping Service has not been set.</td>
</tr>
<tr>
<td>FEDEX_SHIP_SRVC_UNAVAILBL</td>
<td>The FedEx shipping services application is currently unavailable. Please try your request again in a few minutes.</td>
</tr>
<tr>
<td>FEDEX_UNSUPRTD_ORIGIN_COUNTRY</td>
<td>The origin country {1} is currently not supported for Item Fulfillments when using a FedEx shipping method.</td>
</tr>
<tr>
<td>FEDEX_USD_EXCHANGE_RATE_REQD</td>
<td>Cannot retrieve FedEx realtime rates: USD Exchange Rate is required when requesting FedEx realtime rates.</td>
</tr>
<tr>
<td>FEDEX_VOID_ERROR</td>
<td>The FedEx Void failed due to a system error.</td>
</tr>
<tr>
<td>FIELD_CALL_DATE_REQD</td>
<td>Missing Required Field: Call Date</td>
</tr>
<tr>
<td>FIELD_DEFN_REQD</td>
<td>Field definition not found</td>
</tr>
<tr>
<td>FIELD_NOT_SETTABLE_ON_ADD</td>
<td>You are not allowed to set the nsKey for a record</td>
</tr>
<tr>
<td>FIELD_PARAM_REQD</td>
<td>Please enter a value for {1}</td>
</tr>
<tr>
<td>FIELD_PARAM_REQD</td>
<td>Please enter values for {1}.</td>
</tr>
<tr>
<td>FIELD_REQD</td>
<td>Mandatory Field Missing</td>
</tr>
<tr>
<td>FIELD_REQD</td>
<td>You must first select a field</td>
</tr>
<tr>
<td>FILE_ALREADY_EXISTS</td>
<td>A file with the same name already exists in the selected folder.</td>
</tr>
<tr>
<td>FILE_ALREADY_EXISTS</td>
<td>Note: You are attempting to upload a file with a name matching an existing file in the selected folder. Please rename this file or select another folder, and then upload your file.</td>
</tr>
<tr>
<td>FILE_DISALLWD_IN_ROOT_FLDR</td>
<td>You attempted to copy a file to the root directory. Only folders can exist in the root directory.</td>
</tr>
<tr>
<td>FILE_DISALLWD_IN_ROOT_FLDR</td>
<td>You attempted to move a file to the root directory. Only folders can exist in the root directory.</td>
</tr>
<tr>
<td>FILE_MISSING</td>
<td>File Missing</td>
</tr>
<tr>
<td>FILE_NOT_DOWNLOADABLE</td>
<td>Illegal request for a file that isn't downloadable</td>
</tr>
<tr>
<td>FILE_NOT_FOUND</td>
<td>File/Media Item (1) not found.</td>
</tr>
<tr>
<td>FILE_NOT_FOUND</td>
<td>File not found. Please try your download again.</td>
</tr>
<tr>
<td>FILE_REQD</td>
<td>You must enter a file before submitting this form.</td>
</tr>
<tr>
<td>FILE_REQD</td>
<td>You must upload a file before creating this media item</td>
</tr>
<tr>
<td>FILE_UPLOAD_IN_PROGRESS</td>
<td>Files are currently being uploaded to this account.</td>
</tr>
<tr>
<td>FILTER_BY_AMT_REQD</td>
<td>Please enter an amount to filter by.</td>
</tr>
<tr>
<td>FINANCE_CHARGE_SET_PREFS</td>
<td>Finance Charge Item cannot be edited as an item. Please go to the finance charge preferences page to make changes</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>FINANCE_CHARGE_SETUP_REQD</td>
<td>Please set &lt;a href='/app/setup/finchargepref.nl'&gt;Finance Charge Preferences&lt;/a&gt; first.</td>
</tr>
<tr>
<td>FIRST_LAST_NAMES_REQD</td>
<td>Please enter both your first and last name.</td>
</tr>
<tr>
<td>FIRST_QTY_BUCKET_MUST_BE_ZERO</td>
<td>Quantity defined for first quantity bucket must be zero</td>
</tr>
<tr>
<td>FLD_VALUE_REQD</td>
<td>Results are incomplete. You must provide a value for field {1}.</td>
</tr>
<tr>
<td>FLD_VALUE_TOO_LARGE</td>
<td>Value for field {1} is too large to be processed.</td>
</tr>
<tr>
<td>FOLDER_ALREADY_EXISTS</td>
<td>A folder with the same name already exists in the selected folder.</td>
</tr>
<tr>
<td>FORM_RESUBMISSION_REQD</td>
<td>You have logged in to a different user since you navigated to this form. You must re-submit this form as the new user.</td>
</tr>
<tr>
<td>FORM_SETUP_REQD</td>
<td>No appropriate forms are enabled for this role. Please contact your Administrator.</td>
</tr>
<tr>
<td>FORM_UNAVAILBL_ONLINE</td>
<td>This form is not available online.</td>
</tr>
<tr>
<td>FORMULA_ERROR</td>
<td>Your formula has an error in it. It could resolve to the wrong datatype, use an unknown function, or have a syntax error. Please go back, correct the formula, and re-submit.</td>
</tr>
<tr>
<td>FRIENDLY_NAME_REQD</td>
<td>Missing Friendly Name. Friendly Name is a required field and it cannot be null or empty.</td>
</tr>
<tr>
<td>FULFILL_REQD_FIELDS_MISSING</td>
<td>For the listed items, please edit the item record and provide values for the specified fields, and retry the fulfillment.</td>
</tr>
<tr>
<td>FULFILL_REQD_FIELDS_MISSING</td>
<td>The {1} field is required to complete this fulfillment. Please return to the International tab on the item fulfillment and provide a value for the specified field and retry the fulfillment.</td>
</tr>
<tr>
<td>FULFILL_REQD_PARAMS_MISSING</td>
<td>Could not perform operation '{1}' since (2) parameter was not set.</td>
</tr>
<tr>
<td>FULL_DISTRIB_REQD</td>
<td>You must fully distribute all {1} numbers for {1} numbered items.</td>
</tr>
<tr>
<td>FULL_USERS_REQD_TO_INTEGRATE</td>
<td>Only full {1} users can integrate with partners.</td>
</tr>
<tr>
<td>FX_MALFORMED_RESPONSE</td>
<td>Received malformed response from Foreign Exchange source.</td>
</tr>
<tr>
<td>FX_RATE_REQD_FEDEX_RATE</td>
<td>Cannot retrieve {1} realtime rates: {2} Exchange Rate is required when requesting {3} realtime rates.</td>
</tr>
<tr>
<td>FX_TRANS_DISALLWD</td>
<td>FX transactions not accepted for this account. Contact Merchant e-Solutions.</td>
</tr>
<tr>
<td>GETALL_RCRD_TYPE_REQD</td>
<td>The getAll record type is required.</td>
</tr>
<tr>
<td>GIFT_CERT_AMT_EXCEED_AVAILBL</td>
<td>Gift certificate redemption amount exceeds available amount on the gift certificate</td>
</tr>
<tr>
<td>GIFT_CERT_AUTH_ALREADY_EXISTS</td>
<td>Gift certificate authorization code {1} already exists</td>
</tr>
<tr>
<td>GIFT_CERT_CAN_BE_USED_ONCE</td>
<td>A gift certificate may only be used one time on a transaction</td>
</tr>
<tr>
<td>GIFT_CERT_CODE_REQD</td>
<td>Gift certificate codes are missing.</td>
</tr>
<tr>
<td>GIFT_CERT_CODE_REQD</td>
<td>Missing gift certificate authorization code(s). Please go back and enter authorization codes on the {1}.</td>
</tr>
<tr>
<td>GIFT_CERT_CODE_REQD</td>
<td>You must specify a gift certificate code.</td>
</tr>
<tr>
<td>GIFT_CERT_IN_USE</td>
<td>Another user is using gift certificate {1}.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GIFT_CERT_IN_USE</td>
<td>Gift certificate code (1) is already in use</td>
</tr>
<tr>
<td>GIFT_CERT_INVALID_NUM</td>
<td>Gift certificate numbers may not contain the '{1}' character.</td>
</tr>
<tr>
<td>GROUP_DSNT_EXIST</td>
<td>That group does not exist</td>
</tr>
<tr>
<td>GROUP_REQD</td>
<td>You cannot perform a bulk merge operation with an empty group</td>
</tr>
<tr>
<td>GROUP_TYPE_REQD</td>
<td>The group type is required.</td>
</tr>
<tr>
<td>GRTR_QTY_PRICE_LEVEL_REQD</td>
<td>Each quantity pricing level must be greater than the previous quantity pricing level.</td>
</tr>
<tr>
<td>ILLEGAL_ID</td>
<td>Illegal ID. Please enter a name.</td>
</tr>
<tr>
<td>ILLEGAL_PERIOD_STRUCTURE</td>
<td>Illegal period structure. Date (1) is in multiple periods.</td>
</tr>
<tr>
<td>INACTIVE_CC_PROFILE</td>
<td>The credit card processing profile provided is inactive.</td>
</tr>
<tr>
<td>INACTIVE_RCRD_FOR_ROLE</td>
<td>The record for this role has been made inactive.</td>
</tr>
<tr>
<td>INAVLID_FILE_TYP</td>
<td>A change has been made to this file's format. You cannot upload this type of file.</td>
</tr>
<tr>
<td>INAVLID_FILE_TYP</td>
<td>You attempted to upload a restricted file type. Please try again with a selection from the list below:</td>
</tr>
<tr>
<td>INAVLID_ITEM_TYP</td>
<td>Invalid item type [(1)] for item [(2)].</td>
</tr>
<tr>
<td>INAVLID_PRICING_MTRX</td>
<td>Invalid Quantity Pricing Matrix for quantity level (1) : Quantity (2), Base Price (3)</td>
</tr>
<tr>
<td>INCOMPATIBLE_ACCT_CHANGE</td>
<td>The account change you have made is incompatible with old transactions. If you need to swap two accounts, you need to do it in 3 steps. For example, to change the income and asset accounts for an item:&lt;ul&gt;&lt;li&gt;Change the income account to a temporary account and save&lt;/li&gt;&lt;li&gt;Change asset account to the old income account and save&lt;/li&gt;&lt;li&gt;Change the income to the old asset account and save&lt;/ul&gt;Please contact customer support if you need assistance with this.</td>
</tr>
<tr>
<td>INCOMPATIBLE_ACCT_CHANGE</td>
<td>The account change you have made is incompatible with old transactions. Please either change the account selection appropriately or do not request to update past transactions.</td>
</tr>
<tr>
<td>INCOMPLETE_BILLING_ADDR</td>
<td>Billing address is incomplete.</td>
</tr>
<tr>
<td>INCOMPLETE_FILE_UPLOAD</td>
<td>The upload did not complete correctly. Please try uploading the file again. If you have repeatedly received this error message, please send mail to &lt;a href=&quot;mailto:{1}&quot;&gt;{2} NetSuite Customer Support&lt;/a&gt;.</td>
</tr>
<tr>
<td>INCRCRT_ORD_INFO</td>
<td>The order contains incorrect information and was not placed.</td>
</tr>
<tr>
<td>INITIALIZE_ARG_REQD</td>
<td>The initialize reference id is required.</td>
</tr>
<tr>
<td>INITIALIZE_ARG_REQD</td>
<td>The initialize reference type is required.</td>
</tr>
<tr>
<td>INITIALIZE_ARG_REQD</td>
<td>The initialize type is required.</td>
</tr>
<tr>
<td>INITIALIZE_AUXREF_REQD</td>
<td>The initialize auxReference type is required.</td>
</tr>
<tr>
<td>INSTALL_SCRIPT_ERROR</td>
<td>Installation Script Error</td>
</tr>
<tr>
<td>INSUFCNT_NUM_PRDS_FOR_REV_REC</td>
<td>Not enough accounting periods in range specified for revenue recognition.</td>
</tr>
<tr>
<td>INSUFCNT_OPEN_PRDS_FOR_REV_REC</td>
<td>Not enough open accounting periods available for revenue recognition.</td>
</tr>
<tr>
<td>INSUFFICIENT_CHARS_IN_SEARCH</td>
<td>Global searches must contain at least three characters to prevent excessive matches.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>INSUFFICIENT_FLD_PERMISSION</td>
<td>You are attempting to read an unauthorized field: {1}</td>
</tr>
<tr>
<td>INSUFFICIENT_FLD_PERMISSION</td>
<td>You cannot access this search because it includes restricted fields. Please contact your administrator.</td>
</tr>
<tr>
<td>INSUFFICIENT_FUND</td>
<td>Decline. Insufficient funds.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>For security reasons, only an administrator is allowed to edit an administrator record.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>Global search is not permitted from this role.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>Insufficient privileges</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>Your issue DB access has been inactivated. Please contact your issue DB administrator.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>Your current login role does not have an associated Issue Role. Please change to a different role or contact your Issue administrator.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>Only the owner can make a contact private</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>Only the super user can update or delete bug entries</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>Permission error: you may not edit this role.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>Permission Violation: partners do not have access to this report.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>Permission Violation: partners may not delete saved reports.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>Permission Violation: You cannot delete saved reports not created by yourself.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>&lt;b&gt;Test Drive does not allow access to this feature.&lt;/b&gt; If you would like more information about this feature, please contact your account manager.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>The restriction settings on your role deny you access to this item.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>This folder does not exist or you do not have permission to access this folder.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>This folder does not permit the direct addition of files</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>This order has been partially or fully processed and may not be edited by a user without permission to approve sales orders.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>User permission level could not be established</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have permissions to set a value for element {1} due to one of the following reasons: 1) The field is read-only; 2) An associated feature is disabled; 3) The field is available either when a record is created or updated, but not in both cases.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>Your role does not have permission to provision accounts.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You are not allowed to approve your own transactions.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You are not authorized to change this event's organizer. Public events may only have their organizer changed by administrators, the event's organizer, or delegates with edit permission to the event's calendar. Private or busy events may only have their organizer changed by the owner.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You cannot update a system defined template.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You cannot update cases using this form.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You can not access this page unless you are logged in as the consolidated parent company.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You can only delete notes that you created.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have access to the activity history for that record</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have access to the media item you selected.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have access to this page</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have access to this template</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have permission to access this list.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have permission to access this register.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have permission to access this type of transaction.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have permission to create this type of record. Please choose a different record type.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have permission to email transactions.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have permission to perform this operation.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have permission to print (1)</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have permission to view this page.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have privileges to approve commissions.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have privileges to create commissions.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have privileges to create this transaction.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have privileges to perform that operation.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have privileges to perform this action.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have privileges to perform this operation</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have privileges to use this page.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have privileges to view this account</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have privileges to view this page</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You may not create a new Liability Adjustment or edit existing Liability Adjustments.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You may not delete built-in audiences.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You may not delete built-in categories.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You may not delete built-in items.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You may not delete built-in tabs.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You must have either 'Transactions -&gt; Invoice' or 'Transactions -&gt; Cash Sale' permission to bill sales orders.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You must have either 'Transactions -&gt; Invoice' or 'Transactions -&gt; Cash Sale' permission to fulfill sales orders.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You must have 'Transactions -&gt; (1)permission to build work orders.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You must have 'Transactions -&gt; Fulfill Sales Orders' view permission to view sales order fulfillments.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You must have 'Transactions -&gt; Fulfill Sales Orders' edit permission to fulfill sales orders.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You need employee access to delete this record.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>{1} The {2} restrictions on your role deny you access to this record.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>{1} The {2} restrictions on your role prevent you from seeing this record.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>{1} The customer restrictions on your partner role prevent you from seeing this record.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>{1} The restrictions on your role deny you access to this record.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>{1} The restrictions on your role do not allow you to modify this record.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>{1} You need (2) the '{3}' permission to access this page. Please contact your account administrator.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>{1} You need a higher level of the '{2}' permission to access this page. Please contact your account administrator.</td>
</tr>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>{1} You need a higher permission for custom record type {2} to access this page. Please contact your account administrator.</td>
</tr>
<tr>
<td>INTEGER_REQD_FOR_QTY</td>
<td>Quantity must be an integer for numbered items.</td>
</tr>
<tr>
<td>INTL_FEDEX_ONE_PACKG_ALLWD</td>
<td>International FedEx fulfillments allow only one package. If more than one package is required, please break up the shipment into multiple fulfillments of one package each.</td>
</tr>
<tr>
<td>INTL_SHIP_EXCEED_MAX_ITEM</td>
<td>The maximum number of items for FedEx International shipping has been exceeded: {1}</td>
</tr>
<tr>
<td>INVALID_ABN</td>
<td>Invalid ABN registration number {1}.</td>
</tr>
<tr>
<td>INVALID_ACCT</td>
<td>Invalid login. No such account.</td>
</tr>
<tr>
<td>INVALID_ACCT</td>
<td>Invalid account number.</td>
</tr>
<tr>
<td>INVALID_ACCT_NUM_CSTM_FIELD</td>
<td>The account number custom field does not exist!! Consult billing cell.</td>
</tr>
<tr>
<td>INVALID_ACCT_PRD</td>
<td>You can not create an accounting period that is not a year or does not belong to a year.</td>
</tr>
<tr>
<td>INVALID_ACCT_TYP</td>
<td>Invalid account type [{1}].</td>
</tr>
<tr>
<td>INVALID_ACCT_TYP</td>
<td>There is no account of type: {1}</td>
</tr>
<tr>
<td>INVALID_ACCT_TYP</td>
<td>The account and its parent have different account type.</td>
</tr>
<tr>
<td>INVALID_ACCT_TYP</td>
<td>You cannot change an account to or from A/R or A/P</td>
</tr>
<tr>
<td>INVALID_ACTION</td>
<td>You have attempted an unsupported action.</td>
</tr>
<tr>
<td>INVALID_ADDRESS_OR_SHIPPER_NO</td>
<td>An error has occurred. Please ensure that the address information and shipper number are correct, then resubmit the form.</td>
</tr>
<tr>
<td>INVALID_ADJUSTMENT_ACCT</td>
<td>The account you selected in Adjustment Account is the same as the asset account for one of the items you are adjusting. Please go back and change the account. Normally, the adjustment account would be an expense account.</td>
</tr>
<tr>
<td>INVALID_AES_FTSR_EXEMPTN_NUM</td>
<td>The AES/FTSR Exemption Number is invalid.</td>
</tr>
<tr>
<td>INVALID_ALLOCTN_METHOD</td>
<td>You have attempted to allocate landed costs to a transaction using an allocation method that results in no allocation for any lines in the transaction. The allocation method you chose is (1). To correct this problem, go back to the transaction and choose a different allocation method, or modify the items/lines on the transaction so that there will be some cost allocated to the lines.</td>
</tr>
<tr>
<td>INVALID_AMORTZN_ACCT</td>
<td>The destination account for the amortization schedule could not be determined.</td>
</tr>
<tr>
<td>INVALID_AMT</td>
<td>Amount applied greater than total payments and credits.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INVALID_AMT</td>
<td>Foreign currency transactions that use Revenue Commitments cannot have a non-discount line with a negative amount. Please use a non-posting discount item instead.</td>
</tr>
<tr>
<td>INVALID_AMT</td>
<td>Amount Error (credit) Insufficient Funds (debit). Transaction amount is 0 or too long. Resubmit transaction with a valid amount.</td>
</tr>
<tr>
<td>INVALID_AMT</td>
<td>No Action Taken (credit) PIN entry necessary (debit). Reversal amount larger then original amount. No action to take, transaction may not qualify for best level of Interchange.</td>
</tr>
<tr>
<td>INVALID_AMT</td>
<td>Base and consumer amounts are inconsistent with the FX rate. Correct provided amounts.</td>
</tr>
<tr>
<td>INVALID_APP_ID</td>
<td>Invalid application id: {1}</td>
</tr>
<tr>
<td>INVALID_ASSIGN_STATUS_COMBO</td>
<td>Invalid assignee/status combination((1)/(2))</td>
</tr>
<tr>
<td>INVALID_ASSIGN_STATUS_COMBO</td>
<td>Invalid assignee/status combination (assignee {1}, status {2}, issue #{3}). No default owner for issue role?</td>
</tr>
<tr>
<td>INVALID_ASSIGN_STATUS_COMBO</td>
<td>Invalid assignee/status combination ({1},{2})</td>
</tr>
<tr>
<td>INVALID_AUTH</td>
<td>Invalid double authorization. The order is currently authorized for {1}, valid until {2}.</td>
</tr>
<tr>
<td>INVALID_AUTH_CODE</td>
<td>You have entered an invalid authorization code for this campaign email address. Please check the authorization code in the email message, and enter it again.</td>
</tr>
<tr>
<td>INVALID_AUTOAPPLY_VALUE</td>
<td>Ambiguous data: &lt;autoApply&gt; has been selected and lines have been selected in the &lt;applyList&gt; element.</td>
</tr>
<tr>
<td>INVALID_AVS_ADDR</td>
<td>AVS Address Length Error. Please correct the AVS Address and re-submit.</td>
</tr>
<tr>
<td>INVALID_AVS_ADDR</td>
<td>Invalid AVS address or zip data. Please correct the AVS data and re-submit.</td>
</tr>
<tr>
<td>INVALID_AVS_ADDR</td>
<td>AVS Zip Length Error. Please correct the AVS Zip and re-submit.</td>
</tr>
<tr>
<td>INVALID_BALANCE_RANGE</td>
<td>Your balance is not within the allowed range.</td>
</tr>
<tr>
<td>INVALID_BILLING_SCHEDUL</td>
<td>The billing schedule definition is incompatible with this transaction. Please modify the current billing schedule or select a different one.</td>
</tr>
<tr>
<td>INVALID_BILLING_SCHEDUL_DATE</td>
<td>Billing schedules may not extend beyond 500 years from today</td>
</tr>
<tr>
<td>INVALID_BILLING_SCHEDUL_ENTRY</td>
<td>You cannot create a billing schedule with two entries on the same date. Please go back and edit the billing schedule or start date.</td>
</tr>
<tr>
<td>INVALID_BIN_NUM</td>
<td>Bin numbers may not contain the '{1}' character</td>
</tr>
<tr>
<td>INVALID_BOM_QTY</td>
<td>Inventory/Assembly quantities cannot be negative</td>
</tr>
<tr>
<td>INVALID_BOOLEAN_VALUE</td>
<td>Checkbox / boolean data must be either 'T' or 'F'</td>
</tr>
<tr>
<td>INVALID_BUG_NUM</td>
<td>Bug number specified was incorrect. (&quot;{1}&quot; isn't a number.)</td>
</tr>
<tr>
<td>INVALID_CAMPAIGN_CHANNEL</td>
<td>You cannot use this channel to setup this event</td>
</tr>
<tr>
<td>INVALID_CAMPAIGN_GROUP_SIZE</td>
<td>While in {1}, you can only send {2} emails per campaign event. Please modify one or more of your target groups to contain {2} members or less. All campaign emails will be sent to your {1} login email address.</td>
</tr>
<tr>
<td>INVALID_CAMPAIGN_STATUS</td>
<td>You cannot set the status of this campaign event back to 'In Progress' because it already has some activity.</td>
</tr>
<tr>
<td>INVALID_CARD</td>
<td>Missing Card Holder Account Data. Please provide valid card data and re-submit.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INVALID_CARD</td>
<td>Card not in authorizers database.</td>
</tr>
<tr>
<td>INVALID_CARD</td>
<td>The referencing transaction (i.e. reversal request) was not carried out with the same card as the original transaction. Re-submit with original card.</td>
</tr>
<tr>
<td>INVALID_CARD_ID</td>
<td>Invalid Card ID. Provide a valid card ID.</td>
</tr>
<tr>
<td>INVALID_CARD_NUM</td>
<td>Invalid Card Number. Please provide a valid card number and re-submit.</td>
</tr>
<tr>
<td>INVALID_CARD_NUM</td>
<td>Card No. Error. Card number has unknown Bank Identification Number (BIN) or fails check digit edit. Re-try card number again, verify with merchant it is a valid card with proper logos for card types the merchants accept.</td>
</tr>
<tr>
<td>INVALID_CARD_TYP</td>
<td>Card Type Not Accepted. Contact Merchant e-Solutions to add the card type.</td>
</tr>
<tr>
<td>INVALID_CASE_FORM</td>
<td>You cannot create cases using this form.</td>
</tr>
<tr>
<td>INVALID_CATGRY_TAX_AGENCY_REQ</td>
<td>A Vendor must be created in a category with the Tax Agency checkbox checked.</td>
</tr>
<tr>
<td>INVALID_CC_EMAIL_ADDRESS</td>
<td>The email address to CC store orders to is invalid. Please go back and correct it.</td>
</tr>
<tr>
<td>INVALID_CC_NUM</td>
<td>Credit card numbers must contain between 13 and 20 digits.</td>
</tr>
<tr>
<td>INVALID_CC_NUM</td>
<td>Credit card number is not valid. Please check that all digits were entered correctly.</td>
</tr>
<tr>
<td>INVALID_CC_NUM</td>
<td>Credit card number must contain only digits.</td>
</tr>
<tr>
<td>INVALID_CERT</td>
<td>Intercompany Transfer Orders cannot be entered when the Accounting Preferences &gt; Order Management &gt; Use Item Cost as Transfer Cost preference is enabled.</td>
</tr>
<tr>
<td>INVALID_CERT</td>
<td>Invalid CA certificate</td>
</tr>
<tr>
<td>INVALID_CERT</td>
<td>Invalid certificate key</td>
</tr>
<tr>
<td>INVALID_CERT</td>
<td>Invalid certificate</td>
</tr>
<tr>
<td>INVALID_CERT_AUTH</td>
<td>Failed to verify client certificate, send email to {1} for help.</td>
</tr>
<tr>
<td>INVALID_CHARGE_AMT</td>
<td>Charge amount too large. This order can't be charged for more than {1}</td>
</tr>
<tr>
<td>INVALID_CHARS_IN_EMAIL</td>
<td>Email address contains invalid characters.</td>
</tr>
<tr>
<td>INVALID_CHARS_IN_NAME</td>
<td>The From Name field cannot contain apostrophes, quotation marks, commas, or greater than or less than signs.</td>
</tr>
<tr>
<td>INVALID_CHARS_IN_NAME</td>
<td>You cannot use the colon <code>:</code> character in the topic name - please remove it.</td>
</tr>
<tr>
<td>INVALID_CHARS_IN_PARAM_FIELD</td>
<td>The Additional Parameters field can not contain any of the following characters: <code>?\&lt;\&gt;|\/@\#\%\^\&amp;\*\+\,\.;\'&quot;</code>. Please remove them and try again</td>
</tr>
<tr>
<td>INVALID_CHARS_IN_URL</td>
<td>Spaces are not allowed in the {1}url.&lt;p&gt;Examples of a valid {1}url are:&lt;br&gt;&lt;b&gt;<a href="http://www.mydomain.com/image.gif">http://www.mydomain.com/image.gif</a>&lt;/b&gt; or &lt;b&gt;<a href="https://one.two.org/user-name/test.jpg">https://one.two.org/user-name/test.jpg</a>&lt;/b&gt;</td>
</tr>
<tr>
<td>INVALID_CHARS_IN_URL</td>
<td>The URL component you have chosen contains a space or one of the following prohibited character: <code>&amp;\&lt;\&gt;|\/@\#\%\^\&amp;\*\+\,\.;\'&quot;</code>. Please remove them and try again</td>
</tr>
<tr>
<td>INVALID_CHECKOUT_EMAIL</td>
<td>The email address to email checkout errors to is invalid. Please go back and correct it.</td>
</tr>
<tr>
<td>INVALID_CITY</td>
<td>Merchant City Length Error. Reduce the city name length.</td>
</tr>
<tr>
<td>INVALID_COLUMN_NAME</td>
<td>Invalid column name in get_invitem_col_sum_all_locs: {1} [ {2} ]</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INVALID_COLUMN_VALUE</td>
<td>An attempt was made to set a column to an invalid value. Dynamic SQL being executed [{1}]</td>
</tr>
<tr>
<td>INVALID_CONTENT_TYPE</td>
<td>Invalid content type. You can only use application/json, application/xml or text/plain with RESTlets.</td>
</tr>
<tr>
<td>INVALID_COSTING_METHOD</td>
<td>SERIAL and LOT are the only costing methods that may be passed as parameters to this page.</td>
</tr>
<tr>
<td>INVALID_CRNCY_EXCH_RATE</td>
<td>Invalid currency conversion rate.</td>
</tr>
<tr>
<td>INVALID_CRYPT_KEY</td>
<td>{1} is not a valid cryptographic key written as a hexadecimal number.</td>
</tr>
<tr>
<td>INVALID_CSTM_FIELD_DATA_TYP</td>
<td>The customfield [[1]] reference object does not match its data type.</td>
</tr>
<tr>
<td>INVALID_CSTM_FIELD_RCRD_TYP</td>
<td>Invalid custom field record type</td>
</tr>
<tr>
<td>INVALID_CSTM_FIELD_REF</td>
<td>The specified custom field reference {1} is invalid.</td>
</tr>
<tr>
<td>INVALID_CSTM_FORM</td>
<td>{1} is an invalid custom form</td>
</tr>
<tr>
<td>INVALID_CSTM_RCRD_KEY</td>
<td>Invalid custom record key [1].</td>
</tr>
<tr>
<td>INVALID_CSTM_RCRD_QUERY</td>
<td>Invalid custom record object in query.</td>
</tr>
<tr>
<td>INVALID_CSTM_RCRD_TYPE_KEY</td>
<td>Invalid custom record type</td>
</tr>
<tr>
<td>INVALID_CSTM_RCRD_TYPE_KEY</td>
<td>Invalid custom record type key.</td>
</tr>
<tr>
<td>INVALID_CSTM_RCRD_TYPE_KEY</td>
<td>{1} refers to a custom list. To get the contents of this list, use the 'get' or 'getAll' operation with a RecordRef of type 'customList'</td>
</tr>
<tr>
<td>INVALID_CSTM_RCRD_TYP_KEY</td>
<td>Invalid custom record type key in query.</td>
</tr>
<tr>
<td>INVALID_CUR</td>
<td>You have entered an invalid currency symbol or internal ID: {1}.</td>
</tr>
<tr>
<td>INVALID_CURR_CODE</td>
<td>Failed to find a currency code for the currency symbol {1}. Verify your currency symbol is ISO-compliant and re-submit.</td>
</tr>
<tr>
<td>INVALID_CURRENCY_CODE</td>
<td>Failed to find currency code for the requested country code. Check country code and retry request.</td>
</tr>
<tr>
<td>INVALID_CURRENCY_CODE</td>
<td>Request currency code must match FX rate currency code. Retry request with a currency code that matches the FX currency code.</td>
</tr>
<tr>
<td>INVALID_CURRENCY_TYP</td>
<td>Currency Type Not Accepted. Contact Merchant e-Solutions to add the currency type.</td>
</tr>
<tr>
<td>INVALID_CUSTOMER_RCRD</td>
<td>This customer record {1} is not valid. Please create the customer first.</td>
</tr>
<tr>
<td>INVALID_DATA</td>
<td>Invalid data combination, can not set {1} to {2} and {3} to {4}</td>
</tr>
<tr>
<td>INVALID_DATA_FORMAT</td>
<td>Invalid data format. You should return a JavaScript object.</td>
</tr>
<tr>
<td>INVALID_DATA_FORMAT</td>
<td>Invalid data format. You should return TEXT.</td>
</tr>
<tr>
<td>INVALID_DATE</td>
<td>Date Error. Invalid date.</td>
</tr>
<tr>
<td>INVALID_DATE</td>
<td>The date &lt; {1} &gt; is invalid. You must specify a date after &lt; {2} &gt;.</td>
</tr>
<tr>
<td>INVALID_DATE_FORMAT</td>
<td>Date field not in your preferred date format</td>
</tr>
<tr>
<td>INVALID_DATE_RANGE</td>
<td>The date range you specified does not enclose all its child periods.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INVALID_DATE_RANGE</td>
<td>Invalid time range. The (1) &quot;(2)&quot; start time (3)) must be earlier than its end time (4)).</td>
</tr>
<tr>
<td>INVALID_DATE_RANGE</td>
<td>Invalid Date Range - the To Date value must be on or after the From Date value</td>
</tr>
<tr>
<td>INVALID_DATE_RANGE</td>
<td>The date range you specified does not fall inside that of the parent period.</td>
</tr>
<tr>
<td>INVALID_DATE_RANGE</td>
<td>Invalid Deal Range - low must be less than projected and high must be greater than projected.</td>
</tr>
<tr>
<td>INVALID_DATE_RANGE</td>
<td>Invalid Deal Range - low must be less than projected and high must be greater than projected.</td>
</tr>
<tr>
<td>INVALID_DELETE_REF</td>
<td>Either RecordRef or CustomRecordRef should be used for 'delete' operation.</td>
</tr>
<tr>
<td>INVALID_DESTINATION_FOLDER</td>
<td>The destination folder is the same as the current folder.</td>
</tr>
<tr>
<td>INVALID_DESTINATION_COUNTRY</td>
<td>The destination Country is invalid or has not been set.</td>
</tr>
<tr>
<td>INVALID_DESTINATION_POST_CODE</td>
<td>The destination Postal Code is invalid or has not been set.</td>
</tr>
<tr>
<td>INVALID_DESTINATION_STATE</td>
<td>The destination State is invalid or has not been set.</td>
</tr>
<tr>
<td>INVALID_DETACH_RECORD_TYP</td>
<td>Missing or Invalid RecordType for DetachFrom.</td>
</tr>
<tr>
<td>INVALID_DETACH_RECORD_TYP</td>
<td>Detaching of record type {1} from {2} is not supported.</td>
</tr>
<tr>
<td>INVALID_DISCOUNT_MARKUP</td>
<td>Posting and non-posting discounts/markups are not allowed on the same transaction</td>
</tr>
<tr>
<td>INVALID_DOMAIN_KEY</td>
<td>The private domain key is invalid, please enter a valid private domain key.</td>
</tr>
<tr>
<td>INVALID_DOMAIN_NAME</td>
<td>Invalid domain name {1}, please enter a valid domain name.</td>
</tr>
<tr>
<td>INVALID_DUP_ISSUE_REF</td>
<td>Cannot set this issue to be a duplicate of itself or of an issue that is a duplicate of this issue.</td>
</tr>
<tr>
<td>INVALID_EMAIL</td>
<td>Email address is not valid.</td>
</tr>
<tr>
<td>INVALID_EMAIL</td>
<td>Your email or code is invalid. Please try again</td>
</tr>
<tr>
<td>INVALID_EMAIL</td>
<td>You have entered an invalid email address. Please try again.</td>
</tr>
<tr>
<td>INVALID_EMAIL_ADDR</td>
<td>Some of the email addresses you have entered are invalid: {1:list of invalid email addresses}</td>
</tr>
<tr>
<td>INVALID_EMAIL_ADDR</td>
<td>The email address for the web store is invalid. Please go back and correct it.</td>
</tr>
<tr>
<td>INVALID_END_DATE</td>
<td>You entered an end date (1) that is before the start date (2)</td>
</tr>
<tr>
<td>INVALID_END_DATE</td>
<td>{1} [{2}] recurrence end date is invalid</td>
</tr>
<tr>
<td>INVALID_END_TIME</td>
<td>invalid 'end' time</td>
</tr>
<tr>
<td>INVALID_ENTITY_INTERNALID</td>
<td>Attempt to insert entity with nkey -1 or 0</td>
</tr>
<tr>
<td>INVALID_ENTITY_STATUS</td>
<td>You entered an invalid entity status.</td>
</tr>
<tr>
<td>INVALID_EVENT_TIME</td>
<td>You cannot make the time that close to the start or end of the day, because it shifts the event across a day boundary.</td>
</tr>
<tr>
<td>INVALID_EXP_DATE</td>
<td>Invalid expiration date. Please correct the expiration date and re-submit.</td>
</tr>
<tr>
<td>INVALID_EXPNS_ACCT_SUB</td>
<td>The expense account associated with expense category '{1:expense category}' is not available in the subsidiary of customer '{2:customer}'.</td>
</tr>
<tr>
<td>INVALID_EXPRESSION</td>
<td>ERROR: Invalid Expression</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INVALID_FAX_NUM</td>
<td>The Fax Number is invalid.</td>
</tr>
<tr>
<td>INVALID_FAX_PHONE_FORMAT</td>
<td>Invalid FaxPhoneNumber. The format of FaxPhoneNumber must contain area code plus seven digit number.</td>
</tr>
<tr>
<td>INVALID_FIELD_FOR_RCRD_TYP</td>
<td>Record type {1} does not support field {2}</td>
</tr>
<tr>
<td>INVALID_FIELD_NAME_FOR_NULL</td>
<td>The specified name {1} must exactly match an existing field name.</td>
</tr>
<tr>
<td>INVALID_FILE</td>
<td>Verify that you have a valid file to upload.</td>
</tr>
<tr>
<td>INVALID_FILE_ENCODING</td>
<td>The file encoding: {1} is not valid. Please refer to the documentation for a list of supported file encodings.</td>
</tr>
<tr>
<td>INVALID_FILE_TYP</td>
<td>Invalid file type. File is not a compressed/zip file.</td>
</tr>
<tr>
<td>INVALID_FILE_TYP</td>
<td>Invalid file type. File is not a compressed zip file.</td>
</tr>
<tr>
<td>INVALID_FILE_TYP</td>
<td>The media file type you uploaded was not recognized. Please try again.</td>
</tr>
<tr>
<td>INVALID_FLD</td>
<td>ERROR: Field Not Found</td>
</tr>
<tr>
<td>INVALID_FLD_RANGE</td>
<td>Value outside of valid min/max range for this field</td>
</tr>
<tr>
<td>INVALID_FLD_TYPE</td>
<td>Application error: NLField of type {1} is not supported.</td>
</tr>
<tr>
<td>INVALID_FLD_VALUE</td>
<td>You have entered an Invalid Field Value {1} for the following field: {2}</td>
</tr>
<tr>
<td>INVALID_FLD_VALUE</td>
<td>Error in update_folder_size</td>
</tr>
<tr>
<td>INVALID_FORMAT_IN_PARAM_Field</td>
<td>The Additional Parameters field is not formatted correctly. Please reformat and try again</td>
</tr>
<tr>
<td>INVALID_FORMULA</td>
<td>Your formula contains a reference to an encrypted field. This is not allowed.</td>
</tr>
<tr>
<td>INVALID_FULFILMENT_ITEM</td>
<td>You have an invalid item {1} in the fulfillment request.</td>
</tr>
<tr>
<td>INVALID_FX_BASE_CURRENCY</td>
<td>FX amount in base currency is required. Provide the base amount.</td>
</tr>
<tr>
<td>INVALID_FX_RATE</td>
<td>Exchange Rate must be 1 for vendors in your currency.</td>
</tr>
<tr>
<td>INVALID_GET_REF</td>
<td>Either RecordRef or CustomRecordRef should be used for 'get' operation.</td>
</tr>
<tr>
<td>INVALID_GIFT_CERT</td>
<td>Invalid gift certificate</td>
</tr>
<tr>
<td>INVALID_GIFT_CERT_AMT</td>
<td>The remaining amount on a gift certificate cannot be negative</td>
</tr>
<tr>
<td>INVALID_GIFT_CERT_CODE</td>
<td>Gift certificate code must contain only letters and digits.</td>
</tr>
<tr>
<td>INVALID_GROUP_TYP</td>
<td>This type of group cannot be defined based on another group of the same type.</td>
</tr>
<tr>
<td>INVALID_GROUP_TYP</td>
<td>You cannot define this group type using this search.</td>
</tr>
<tr>
<td>INVALID_GROUP_TYPE</td>
<td>The group type {1} is invalid.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INVALID_GRP</td>
<td>This type of group cannot be defined based on another group.</td>
</tr>
<tr>
<td>INVALID_GST_PST_AGENCIES</td>
<td>The GST or PST agencies are not valid. Please review your company preferences.</td>
</tr>
<tr>
<td>INVALID_ID</td>
<td>No order found with id {1}.</td>
</tr>
<tr>
<td>INVALID_ID</td>
<td>Identifiers can contain only digits, alphabetic characters, or &quot;_&quot; with no spaces.</td>
</tr>
<tr>
<td>INVALID_ID</td>
<td>You have provided an invalid script id or internal id: {1}.</td>
</tr>
<tr>
<td>INVALID_ID</td>
<td>The externalId attribute is not supported for {1}.</td>
</tr>
<tr>
<td>INVALID_INITIALIZE_ARG</td>
<td>The reference type {1} and initialize type {2} are not matched.</td>
</tr>
<tr>
<td>INVALID_INITIALIZE_ARG</td>
<td>InitializeRef should be used for 'initialize' operation.</td>
</tr>
<tr>
<td>INVALID_INITIALIZE_AUXREF</td>
<td>Invalid initialize operation argument 'auxReference'.</td>
</tr>
<tr>
<td>INVALID_INITIALIZE_REF</td>
<td>You can not initialize {1}: invalid reference {2}.</td>
</tr>
<tr>
<td>INVALID_INITIALIZE_REF</td>
<td>Can not initialize customerPayment: invalid customer reference {1}.</td>
</tr>
<tr>
<td>INVALID_INITIALIZE_REF</td>
<td>Can not initialize customerPayment: invalid invoice reference {1}.</td>
</tr>
<tr>
<td>INVALID_INITIALIZE_REF</td>
<td>You have an invalid sales order {1} or the order is already billed.</td>
</tr>
<tr>
<td>INVALID_INITIALIZE_REF</td>
<td>You have an invalid sales order {1} or the order is already closed.</td>
</tr>
<tr>
<td>INVALID_INSURED_VALUE</td>
<td>The Insured Value cannot exceed the total sum of the items being shipped.</td>
</tr>
<tr>
<td>INVALID_INTERNAL_ID</td>
<td>The specified internal id is not allowed.</td>
</tr>
<tr>
<td>INVALID_INTERNALID</td>
<td>Unparseable Internal Id, did you mean to lookup this field by Name or External ID?</td>
</tr>
<tr>
<td>INVALID_INV_DATE</td>
<td>Invoice date on billing schedule may not be after {1}.</td>
</tr>
<tr>
<td>INVALID_INVENTORY_NUM</td>
<td>Invalid set of inventory numbers: values must be separated by commas, spaces, tabs, or line feeds.</td>
</tr>
<tr>
<td>INVALID_INV_DATE</td>
<td>Invoice date on billing schedule may not be after {1}.</td>
</tr>
<tr>
<td>INVALID_IP_ADDRESS_RULE</td>
<td>The following IP Address rule is not valid: {1}.</td>
</tr>
<tr>
<td>INVALID_ISSUE_BUILD_VERSION</td>
<td>Cannot set issue {1} to (2) (3) and {4} {5} because that version is not associated with that build.</td>
</tr>
<tr>
<td>INVALID_ISSUE_PRIORITY</td>
<td>Severity 1 issues must have priority 1.</td>
</tr>
<tr>
<td>INVALID_ISSUE_PRODUCT</td>
<td>Cannot set issue {1} to (2) (3) and {4} {5} because that product is not associated with that build.</td>
</tr>
<tr>
<td>INVALID_ISSUE_PRODUCT</td>
<td>Cannot set issue {1} to (2) (3) and {4} {5} because that product is not associated with that module.</td>
</tr>
<tr>
<td>INVALID_ISSUE_STATUS</td>
<td>Cannot set issue {1} to status {2} and assignee {3} because that status requires an assignee with issue role {4}.</td>
</tr>
<tr>
<td>INVALID_ITEM_OPTION</td>
<td>Invalid item option {1} for item {2}.</td>
</tr>
<tr>
<td>INVALID_ITEM_OPTIONS</td>
<td>The options for item {'1'} are no longer available. Please change your order and try again.</td>
</tr>
<tr>
<td>INVALID_ITEM_SUBTYP</td>
<td>Invalid item subtype [{1}] for item [{2}].</td>
</tr>
<tr>
<td>INVALID_ITEM_TYP</td>
<td>The item [{1}] does not have a valid item type.</td>
</tr>
<tr>
<td>INVALID_ITEM_WEIGHT</td>
<td>The total item weight must be &gt; 0.0.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INVALID_JOB_ID</td>
<td>You have specified an invalid Job Id</td>
</tr>
<tr>
<td>INVALID_KEY_OR_REF</td>
<td>The specified key is invalid.</td>
</tr>
<tr>
<td>INVALID_KEY_OR_REF</td>
<td>Invalid (1) reference key (2).</td>
</tr>
<tr>
<td>INVALID_KEY_OR_REF</td>
<td>Invalid (1) reference key (2) for (3) (4).</td>
</tr>
<tr>
<td>INVALID_KEY_PASSWORD_REQD</td>
<td>This key is invalid or may require a password</td>
</tr>
<tr>
<td>INVALID_LINE_ID</td>
<td>No line items match the entered id(s) (1).</td>
</tr>
<tr>
<td>INVALID_LINK_SUM</td>
<td>Links sum to more than applied transaction amount</td>
</tr>
<tr>
<td>INVALID_LINK_SUM</td>
<td>Links sum to more than original transaction amount</td>
</tr>
<tr>
<td>INVALID_LIST_ID</td>
<td>You must specify a valid line ID. Please set (1).</td>
</tr>
<tr>
<td>INVALID_LIST_KEY</td>
<td>Could not perform operation &quot;(1)&quot; on an invalid line [(2)].</td>
</tr>
<tr>
<td>INVALID_LIST_KEY</td>
<td>Could not perform operation 'add' on an existing line [(1)].</td>
</tr>
<tr>
<td>INVALID_LOC</td>
<td>Item Fulfillment/Item Receipt location does not match the location on the Transfer Order</td>
</tr>
<tr>
<td>INVALID_LOC_SUB_RESTRICTN</td>
<td>You may not add inventory to a location that is incompatible with the subsidiary restrictions for this item.</td>
</tr>
<tr>
<td>INVALID_LOGIN</td>
<td>Invalid login. Online Form access is disabled.</td>
</tr>
<tr>
<td>INVALID_LOGIN</td>
<td>Invalid login. Supplier access is disabled.</td>
</tr>
<tr>
<td>INVALID_LOGIN_ATTEMPT</td>
<td>Invalid login attempt.</td>
</tr>
<tr>
<td>INVALID_LOGIN_CREDENTIALS</td>
<td>A problem occurred verifying the presented email address, password, roleName or account number, please verify these pieces of information and try again</td>
</tr>
<tr>
<td>INVALID_LOGIN_CREDENTIALS</td>
<td>You have entered an invalid email address or account number. Please try again.</td>
</tr>
<tr>
<td>INVALID_LOGIN_CREDENTIALS</td>
<td>You have entered an invalid email address or password. Please try again.</td>
</tr>
<tr>
<td>INVALID_LOGIN_CREDENTIALS</td>
<td>You have entered an invalid login password. Please try again.</td>
</tr>
<tr>
<td>INVALID_LOGIN_CREDENTIALS</td>
<td>You have entered an invalid login password. Please try again.</td>
</tr>
<tr>
<td>INVALID_LOGIN_IP</td>
<td>Invalid login. IP Address does not match any of the IP Address rules specified for this entity.</td>
</tr>
<tr>
<td>INVALID_LOT_NUM_FORMAT</td>
<td>Lot numbers must be entered using this format: LOT#(Quantity).nFor example, to enter a quantity of 100 items as Lot number ABC1234, enter &quot;ABC1234(100)&quot; in the Lot Numbers field.</td>
</tr>
<tr>
<td>INVALID_MACRO_ID</td>
<td>The Macro ID: (1) is not valid! Please refer to the documentation for a list of supported macro IDs.</td>
</tr>
<tr>
<td>INVALID_MARKUP_DISCOUNT</td>
<td>Markup/Discount % must be between -999% and 999%</td>
</tr>
<tr>
<td>INVALID_MCC</td>
<td>Merchant Category Code Length Error. Provide a valid MCC.</td>
</tr>
<tr>
<td>INVALID_MEMBER_HIERARCHY</td>
<td>You have defined a group/kit/assembly item that contains a loop in the member hierarchy. You must remove any group/kit/assembly member items that contain this item as a member.</td>
</tr>
<tr>
<td>INVALID_MEMRZD_TRANS</td>
<td>A memorized transaction may not contain any serial or lot numbers. Go back, remove the numbers, and try to re-Memorize the transaction. Posting transactions such as Bills or Cash Sales may not use serial or lot numbered items. Non-Posting transactions such as Purchase</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>INVALID_MERCHANT_KEY</td>
<td>Merchant key is not supplied or incorrect.</td>
</tr>
<tr>
<td>INVALID_MERCHANT_NAME</td>
<td>Merchant Name Length Error. Reduce the merchant name length.</td>
</tr>
<tr>
<td>INVALID_NAME</td>
<td>Invalid savepoint name. Must start with an alphabet character and can only contain alphanumeric, underscore, dollar, and hash characters.</td>
</tr>
<tr>
<td>INVALID_NEXUS</td>
<td>Transaction Nexus is incorrect: it is {1} but should be {2}</td>
</tr>
<tr>
<td>INVALID_NUM</td>
<td>Invalid Decimal Number</td>
</tr>
<tr>
<td>INVALID_NUMBER</td>
<td>Invalid Decimal Number</td>
</tr>
<tr>
<td>INVALID_NUMBER</td>
<td>Invalid Integer</td>
</tr>
<tr>
<td>INVALID_NUMBER</td>
<td>Invalid integer {1}</td>
</tr>
<tr>
<td>INVALID_NUMBER</td>
<td>Invalid number {1}</td>
</tr>
<tr>
<td>INVALID_NUMBER</td>
<td>You entered &quot;1&quot; into a field where a numeric value was expected. Please go back and change this value to a number.</td>
</tr>
<tr>
<td>INVALID_NUMBER</td>
<td>You entered an invalid number: Go back, change this value and resubmit.</td>
</tr>
<tr>
<td>INVALID_OBJ</td>
<td>There are no objects of this type</td>
</tr>
<tr>
<td>INVALID_ONLINE_FORM</td>
<td>Online Form not found</td>
</tr>
<tr>
<td>INVALID_ONLINE_FORM</td>
<td>This online form is inactive or not available online</td>
</tr>
<tr>
<td>INVALID_ONLINE_FORM_URL</td>
<td>You cannot submit Online forms from this URL. Use the live version: {1} instead</td>
</tr>
<tr>
<td>INVALID_OPENID_DOMAIN</td>
<td>This is not a valid domain. Please go back and enter your domain name without prefixes such as 'http://' or 'www'.</td>
</tr>
<tr>
<td>INVALID_OPERATION</td>
<td>That operation is not supported for this record type: {1}</td>
</tr>
<tr>
<td>INVALID_ORD_STATUS</td>
<td>This order has been partially or fully processed and may not be reset to 'Pending Approval'.</td>
</tr>
<tr>
<td>INVALID_ORIGIN_COUNTRY</td>
<td>The origin Country is invalid or has not been set.</td>
</tr>
<tr>
<td>INVALID_ORIGIN_POSTCODE</td>
<td>The origin Postal Code is invalid or has not been set.</td>
</tr>
<tr>
<td>INVALID_ORIGIN_STATE</td>
<td>The origin State is invalid or has not been set.</td>
</tr>
<tr>
<td>INVALID_PAGE_INDEX</td>
<td>Job (1) does not have a page (2)</td>
</tr>
<tr>
<td>INVALID_PAGE_PARAM</td>
<td>Invalid page parameter. Unable to view page.</td>
</tr>
<tr>
<td>INVALID_PAGER_NUM</td>
<td>The Pager Number is invalid.</td>
</tr>
<tr>
<td>INVALID_PARAM</td>
<td>Please select either (1) or (2) parameter but not both.</td>
</tr>
<tr>
<td>INVALID_PARENT</td>
<td>An account cannot be its own parent</td>
</tr>
<tr>
<td>INVALID_PARTNER_CODE</td>
<td>An account for this customer cannot be provisioned unless its partnercode ((1)) is empty or numeric.</td>
</tr>
<tr>
<td>INVALID_PARTNER_ID</td>
<td>Invalid partner id: {1}</td>
</tr>
<tr>
<td>INVALID_PASSWORD</td>
<td>Invalid key password</td>
</tr>
<tr>
<td>INVALID_PAYCHECK_DATE</td>
<td>Paychecks for (1) must be on or after (2)</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INVALID_PERIOD</td>
<td>A period may be only an adjustment period, a quarter, or a year.</td>
</tr>
<tr>
<td>INVALID_PHONE</td>
<td>The phone number of the (1) address is invalid. Please verify the phone number is correctly formatted and includes the area code.</td>
</tr>
<tr>
<td>INVALID_PHONE_FAX_PAGER_NUM</td>
<td>The Phone, Fax, or Pager Number is invalid.</td>
</tr>
<tr>
<td>INVALID_PHONE_NUM</td>
<td>The Phone Number is invalid.</td>
</tr>
<tr>
<td>INVALID_PICKUP_POSTAL_CODE</td>
<td>An error has occurred. Pickup Postal Code (1) is not the postal code associated with Shipper Number (2).</td>
</tr>
<tr>
<td>INVALID_PIN</td>
<td>Incorrect PIN. PIN number may have been entered incorrectly. Re-submit with proper PIN.</td>
</tr>
<tr>
<td>INVALID_PIN_DEBIT_TRANS_TYP</td>
<td>Invalid pin debit transaction type. Only a sale (D) is supported.</td>
</tr>
<tr>
<td>INVALID_PORTLET_TYP</td>
<td>unsupported portlet type [(1)], id [(2)] processed by cardMetaDataGenerator</td>
</tr>
<tr>
<td>INVALID_POST</td>
<td>Invalid Post</td>
</tr>
<tr>
<td>INVALID_PRESENTATION_TYP</td>
<td>Presentation Type not recognized</td>
</tr>
<tr>
<td>INVALID_PROBABILITY_RANGE</td>
<td>Probability must be between 0 and 100.</td>
</tr>
<tr>
<td>INVALID_PROFILE_ID</td>
<td>Invalid Profile ID or Profile Key. Correct the profile ID and profile key, and re-submit.</td>
</tr>
<tr>
<td>INVALID_PROJ_BILLING_TYP</td>
<td>The project billing type is incompatible with the billing schedule on the transaction. Please select a different billing schedule.</td>
</tr>
<tr>
<td>INVALID_PST_TAX_VALUE</td>
<td>PST tax value is not a valid number: (1).</td>
</tr>
<tr>
<td>INVALID_PSWD</td>
<td>Email address &quot;(1)&quot; has been previously registered under a different password from the new password you just provided. For security reasons, you will first need to go back and supply the correct new password for &quot;(1)&quot; to merge the accounts.</td>
</tr>
<tr>
<td>INVALID_PSWD</td>
<td>Invalid Password. The password must be between 6 and 10 character with at least one numeric and one alphabetic character.</td>
</tr>
<tr>
<td>INVALID_PSWD</td>
<td>Password must be at least 6 characters long.</td>
</tr>
<tr>
<td>INVALID_PSWD</td>
<td>Password must be at least 6 characters long and contain at least one number or special character.</td>
</tr>
<tr>
<td>INVALID_PSWD</td>
<td>Password must contain at least one letter (A-Z).</td>
</tr>
<tr>
<td>INVALID_PSWD</td>
<td>Password must contain at least one number or special character.</td>
</tr>
<tr>
<td>INVALID_PSWD</td>
<td>The current password you supplied is incorrect.</td>
</tr>
<tr>
<td>INVALID_PSWD</td>
<td>Your new password must be at least (1) characters, contain at least one non-letter, and be substantially different from the current password.</td>
</tr>
<tr>
<td>INVALID_PSWD</td>
<td>Your new password must be at least 6 characters, contain at least one non-letter, and be substantially different from the current password.</td>
</tr>
<tr>
<td>INVALID_PSWD</td>
<td>Your password cannot be the same as your login. Please choose a new password.</td>
</tr>
<tr>
<td>INVALID_PSWD</td>
<td>Your password must be at least 6 characters</td>
</tr>
<tr>
<td>INVALID_PSWD</td>
<td>You've used that password before. Please choose a new password.</td>
</tr>
<tr>
<td>INVALID_PSWD_HINT</td>
<td>Your hint is too similar to your password. Please choose something less obvious.</td>
</tr>
<tr>
<td>INVALID_PSWD_ILLEGAL_CHAR</td>
<td>Password contains an illegal character.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>INVALID_PURCHASE_TAX_CODE</td>
<td>Purchase tax code not defined properly for item</td>
</tr>
<tr>
<td>INVALID_QTY</td>
<td>The new quantity and new value must be either both positive or both negative.</td>
</tr>
<tr>
<td>INVALID_QTY</td>
<td>You may not receive a larger quantity than you shipped.</td>
</tr>
<tr>
<td>INVALID_QUANTITY</td>
<td>Serial and lot number quantities must be integers</td>
</tr>
<tr>
<td>INVALID_QUANTITY</td>
<td>Serial and lot number quantities must be positive.</td>
</tr>
<tr>
<td>INVALID_QUESTION</td>
<td>Please select a different question.</td>
</tr>
<tr>
<td>INVALID_QUESTION</td>
<td>Please select a question.</td>
</tr>
<tr>
<td>INVALID_RCRD</td>
<td>Invalid record specification: {1}</td>
</tr>
<tr>
<td>INVALID_RCRD_CONVERSION</td>
<td>Only customer records can be converted to child or parent records. Please select only customer records for this duplicate merge operation.</td>
</tr>
<tr>
<td>INVALID_RCRD_HEADER_</td>
<td>Invalid record header: Unable to parse field name from {1}</td>
</tr>
<tr>
<td>INVALID_RCRD_HEADER_</td>
<td>Invalid record header: Unable to parse record id from {1}</td>
</tr>
<tr>
<td>INVALID_RCRD_HEADER_</td>
<td>Invalid record header: Unable to parse record type from {1}</td>
</tr>
<tr>
<td>INVALID_RCRD_ID</td>
<td>Invalid id {1} to create a record.</td>
</tr>
<tr>
<td>INVALID_RCRD_INITIALIZE</td>
<td>You have entered an invalid default value for this record initialize operation.</td>
</tr>
<tr>
<td>INVALID_RCRD_OBJ</td>
<td>You do not have a valid record object.</td>
</tr>
<tr>
<td>INVALID_RCRD_REF</td>
<td>Invalid RecordRef internalId {1} for field {2}</td>
</tr>
<tr>
<td>INVALID_RCRD_REF</td>
<td>Invalid record reference.</td>
</tr>
<tr>
<td>INVALID_RCRD_REF</td>
<td>Invalid record reference</td>
</tr>
<tr>
<td>INVALID_RCRD_TRANSFRM</td>
<td>You have entered an invalid default value for this record transformation operation.</td>
</tr>
<tr>
<td>INVALID_RCRD_TRANSFRM</td>
<td>That type of record transformation is not allowed. Please see the documentation for a list of supported transformation types</td>
</tr>
<tr>
<td>INVALID_RCRD_TRANSFRM</td>
<td>That is not a valid record transformation.</td>
</tr>
<tr>
<td>INVALID_RCRD_TYPE</td>
<td>Invalid Record Type</td>
</tr>
<tr>
<td>INVALID_RCRD_TYPE</td>
<td>{1}: type argument (2) is not a valid record or is not available in your account. Please see the documentation for a list of supported record types.</td>
</tr>
<tr>
<td>INVALID_RCRD_TYPE</td>
<td>The record type [{1}] is invalid.</td>
</tr>
<tr>
<td>INVALID_RCRD_TYPE</td>
<td>The record type is invalid.</td>
</tr>
<tr>
<td>INVALID_RECIPIENT</td>
<td>Recipient internal id does not match an existing entity.</td>
</tr>
<tr>
<td>INVALID_RECR_REF</td>
<td>Could not update {1} to {2} because referenced record does not exist</td>
</tr>
<tr>
<td>INVALID_RECUR_DATE_RANGE</td>
<td>This event recurrence is invalid because its duration is either negative or longer than one day. {1}</td>
</tr>
<tr>
<td>INVALID_RECUR_DATE_RANGE</td>
<td>This event recurrence is invalid because its end-by date is before its start date. {1}</td>
</tr>
<tr>
<td>INVALID_RECUR_DATE_RANGE</td>
<td>This event recurrence is invalid because its end time and duration do not match. {1}</td>
</tr>
<tr>
<td>INVALID_RECUR_DATE_RANGE</td>
<td>This event recurrence is invalid because its end time is more than one day after its start time. {1}</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INVALID_RECUR_DATE_RANGE</td>
<td>This event recurrence is invalid because its end time is not after its start time. (1)</td>
</tr>
<tr>
<td>INVALID_RECUR_DATE_RANGE</td>
<td>This event recurrence is invalid because its start time or end time/duration is empty. (1)</td>
</tr>
<tr>
<td>INVALID_RECUR_DATE_RANGE</td>
<td>This event recurrence is invalid because its times are not in order. (1)</td>
</tr>
<tr>
<td>INVALID_RECUR_DESC_REQD</td>
<td>This event recurrence is invalid because it has no description. (1)</td>
</tr>
<tr>
<td>INVALID_RECUR_DOW</td>
<td>This event recurrence has an invalid day-of-week field. (1)</td>
</tr>
<tr>
<td>INVALID_RECUR_DOWIM</td>
<td>This event recurrence has an invalid day-of-week-in-month value. (1)</td>
</tr>
<tr>
<td>INVALID_RECUR_DOWMASK</td>
<td>This event recurrence is invalid because its day-of-week mask is not 7 characters long. (1)</td>
</tr>
<tr>
<td>INVALID_RECUR_FREQUENCY</td>
<td>This event recurrence has an invalid frequency. (1)</td>
</tr>
<tr>
<td>INVALID_RECUR_PATTERN</td>
<td>This event does not have a valid recurrence pattern.</td>
</tr>
<tr>
<td>INVALID_RECUR_PATTERN</td>
<td>This event recurrence is invalid because it is not a monthly or yearly event and it has day-of-week and day-of-week-in-month field values. (1)</td>
</tr>
<tr>
<td>INVALID_RECUR_PATTERN</td>
<td>This event recurrence is invalid because it only has one recurrence time and it must have either none or at least two. (1)</td>
</tr>
<tr>
<td>INVALID_RECUR_PATTERN</td>
<td>This event recurrence is invalid because one of its times is out of the range 0 to 86399. (1)</td>
</tr>
<tr>
<td>INVALID_RECUR_PATTERN</td>
<td>This event recurrence is invalid because only one of the day-of-week and day-of-week-in-month fields is set. Both must be set or both must be unset. (1)</td>
</tr>
<tr>
<td>INVALID_RECUR_PATTERN</td>
<td>This event recurrence is invalid either because it is not weekly and it has a day-of-week mask, or it is weekly and it has no day-of-week mask. (1)</td>
</tr>
<tr>
<td>INVALID_RECUR_PATTERN</td>
<td>This single day event is invalid since it contains a recurrence pattern.</td>
</tr>
<tr>
<td>INVALID_RECUR_PERIOD</td>
<td>This event recurrence has an invalid period. (1)</td>
</tr>
<tr>
<td>INVALID_RECUR_TIME_ZONE_REQD</td>
<td>This event recurrence is invalid because it has no time zone. (1)</td>
</tr>
<tr>
<td>INVALID_REF_CANT_INITIALIZE</td>
<td>Cannot initialize customerRefund: invalid creditMemo reference (1).</td>
</tr>
<tr>
<td>INVALID_REF_CANT_INITIALIZE</td>
<td>Cannot initialize customerRefund: invalid customer reference (1).</td>
</tr>
<tr>
<td>INVALID_REF_CANT_INITIALIZE</td>
<td>You can not initialize (1) by referencing (2).</td>
</tr>
<tr>
<td>INVALID_REF_KEY</td>
<td>Invalid externalId (1).</td>
</tr>
<tr>
<td>INVALID_REF_KEY</td>
<td>Invalid reference key [{1}].</td>
</tr>
<tr>
<td>INVALIDREFERER_EMAIL</td>
<td>The referer email address you have entered is not valid. Please try again.</td>
</tr>
<tr>
<td>INVALID_REFUND_AMT</td>
<td>Refund amount must be between zero and the original amount. Correct amount and retry request.</td>
</tr>
<tr>
<td>INVALID_REFUND_AMT</td>
<td>The amount you can refund is (1) because the the order has already been refunded for (2)</td>
</tr>
<tr>
<td>INVALID_REPORT</td>
<td>The referenced Report and Row are no longer valid, because the layout containing them has changed. Please edit this reference row to reselect Report and Row.</td>
</tr>
<tr>
<td>INVALID_REPORT_ID</td>
<td>Invalid report ID.</td>
</tr>
<tr>
<td>INVALID_REPORT_ROW</td>
<td>Invalid Reference Row</td>
</tr>
<tr>
<td>INVALID_REPORT_ROW</td>
<td>Invalid Report Row Reference</td>
</tr>
<tr>
<td>INVALID_REQUEST</td>
<td>invalid request (failed isValid() check). Email request handler unable to service request for address= &lt;(1)&gt; .</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INVALID_RESOURCE_TIME</td>
<td>Total resource time for '{1}' cannot exceed {2} planned time entries.</td>
</tr>
<tr>
<td>INVALID_RESULT_SUMMARY_FUNC</td>
<td>The result field {1} cannot be grouped. Please edit the search and omit this field or use a different summary function.</td>
</tr>
<tr>
<td>INVALID_RETURN_DATA_OBJECT</td>
<td>The return value of the RESTlet function cannot be used to create a well-formed XML document.</td>
</tr>
<tr>
<td>INVALID_RETURN_DATA_OBJECT</td>
<td>The XML document does not contain a root element because the RESTlet function does not return a JavaScript Object.</td>
</tr>
<tr>
<td>INVALID_REV_REC_DATE_RANGE</td>
<td>Rev rec end date can not be before rev rec start date.</td>
</tr>
<tr>
<td>INVALID_ROLE</td>
<td>The specified role is invalid.</td>
</tr>
<tr>
<td>INVALID_ROLE</td>
<td>Your role does not give you permission to view this page.</td>
</tr>
<tr>
<td>INVALID_ROLE_FOR_EVENT</td>
<td>You seem to have been invited to this {1} in a different role. Please change your role to view the {1}.</td>
</tr>
<tr>
<td>INVALID_RQST_CONTACTS_EXIST</td>
<td>it has associated primary contacts.</td>
</tr>
<tr>
<td>INVALID_RQST_PARENT_REQD</td>
<td>it has associated contact records that would be left with no parent company.</td>
</tr>
<tr>
<td>INVALID_RQST_SBCUST_JOBS_EXIST</td>
<td>it has associated sub-customers or jobs.</td>
</tr>
<tr>
<td>INVALID_SAVED_SRCH</td>
<td>Missing or invalid saved search for Custom KPI. The search must have a date column as an available filter. Please see help next to Custom KPI dropdown list on KPI setup page.</td>
</tr>
<tr>
<td>INVALID_SAVEDSEARCH</td>
<td>A saved search with the internal ID {1} does not exist.</td>
</tr>
<tr>
<td>INVALID_SAVEDSEARCH</td>
<td>We cannot return search columns for summary saved search {1}.</td>
</tr>
<tr>
<td>INVALID_SCHDUL_AMT</td>
<td>The total amount on the schedule must equal the sum of the individual recognition amounts.</td>
</tr>
<tr>
<td>INVALID_SCHDUL_AMT</td>
<td>The total amount on the schedule must be equal to the amount of the source transaction line.</td>
</tr>
<tr>
<td>INVALID_SCHDUL_FORMAT</td>
<td>To create a valid schedule, please enter the bracket values in ascending orders without gaps.</td>
</tr>
<tr>
<td>INVALID_SCRIPT_ID</td>
<td>A saved search with the script ID {1} does not exist.</td>
</tr>
<tr>
<td>INVALID_SEARCH</td>
<td>That search or mass update does not exist.</td>
</tr>
<tr>
<td>INVALID_SEARCH</td>
<td>You may search by {1} or {2} but not both</td>
</tr>
<tr>
<td>INVALID_SEARCH_CRITERIA</td>
<td>Can't search transactions: invalid cross reference key</td>
</tr>
<tr>
<td>INVALID_SEARCH_CRITERIA</td>
<td>Global Search supports at most three keywords and requires at least one. Keywords are composed of only letters, digits, and dashes.</td>
</tr>
<tr>
<td>INVALID_SEARCH_FIELD_KEY</td>
<td>search field keys are not consistent((1)/(2))</td>
</tr>
<tr>
<td>INVALID_SEARCH_FIELD_NAME</td>
<td>search field names are not consistent((1)/(2))</td>
</tr>
<tr>
<td>INVALID_SEARCH_FIELD_OBJ</td>
<td>{1} is not a valid search custom field</td>
</tr>
<tr>
<td>INVALID_SEARCH_FIELD_OBJ</td>
<td>{1} must be used to search custom field (2)</td>
</tr>
<tr>
<td>INVALID_SEARCH_FIELD_OBJ</td>
<td>Server application error: invalid search customfield object.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INVALID_SEARCH_FIELD_OBJ</td>
<td>Invalid search field object: {1}</td>
</tr>
<tr>
<td>INVALID_SEARCH_JOIN_ID</td>
<td>Invalid Search Join ID</td>
</tr>
<tr>
<td>INVALID_SEARCH_MORE</td>
<td>Invalid searchMore operation. Please make sure that you have had a successful search</td>
</tr>
<tr>
<td></td>
<td>operation before you can perform any searchMore operation.</td>
</tr>
<tr>
<td>INVALID_SEARCH_OPERATOR</td>
<td>You need to provide a valid search field operator.</td>
</tr>
<tr>
<td>INVALID_SEARCH_OPERATOR</td>
<td>You can not use this operator '{1}' for internalId search.</td>
</tr>
<tr>
<td>INVALID_SEARCH_PAGE_INDEX</td>
<td>Invalid search page index.</td>
</tr>
<tr>
<td>INVALID_SEARCH_PAGE_SIZE</td>
<td>Invalid search page size.</td>
</tr>
<tr>
<td>INVALID_SEARCH_PREF</td>
<td>You cannot set returnSearchColumns to false while you specify search columns.</td>
</tr>
<tr>
<td>INVALID_SEARCH_PREF</td>
<td>You cannot set returnSearchColumns to true without specifying search columns or referencing</td>
</tr>
<tr>
<td></td>
<td>a saved search.</td>
</tr>
<tr>
<td>INVALID_SEARCH_SELECT_OBJ</td>
<td>Invalid search select field object: {1}</td>
</tr>
<tr>
<td>INVALID_SEARCH_VALUE</td>
<td>You need to provide a search value.</td>
</tr>
<tr>
<td>INVALID_SEARCH_VALUE</td>
<td>You need to provide search values.</td>
</tr>
<tr>
<td>INVALID_SECONDARY_EMAIL</td>
<td>Invalid secondary email address. The email address must be in a valid format.</td>
</tr>
<tr>
<td>INVALID_SECPAY_CREDENTIALS</td>
<td>The username or password used to process the transaction with SECPay was not valid.</td>
</tr>
<tr>
<td></td>
<td>Please make sure you have entered the correct username, password, and remote password in</td>
</tr>
<tr>
<td></td>
<td>your SECPay account setup.</td>
</tr>
<tr>
<td>INVALID_SERIAL_NUM</td>
<td>No items match the entered serial number</td>
</tr>
<tr>
<td>INVALID_SERIAL_OR_LOT_NUMBER</td>
<td>Serial and lot numbers may not contain the '{1}' character.</td>
</tr>
<tr>
<td>INVALID_SESSION</td>
<td>A valid NLSession is required to generate record xml</td>
</tr>
<tr>
<td>INVALID_SESSION</td>
<td>A valid session is required. Please log in first.</td>
</tr>
<tr>
<td>INVALID_SHIP_DATE</td>
<td>The Future Ship Date is invalid. Please verify the entered Future Ship Date is no more than</td>
</tr>
<tr>
<td></td>
<td>7 days in the future, and resubmit the fulfillment.</td>
</tr>
<tr>
<td>INVALID_SHIP_FROM_STATE</td>
<td>The Ship From State/Province Code is missing or invalid. Please enter the 2 to 5 character</td>
</tr>
<tr>
<td></td>
<td>abbreviation for the state or province of the address that contains it.</td>
</tr>
<tr>
<td>INVALID_SHIP_GRP</td>
<td>You cannot add shipping groups when creating a transaction that has multiple shipping</td>
</tr>
<tr>
<td></td>
<td>routes enabled. You must first add the items, then get the transaction and update the</td>
</tr>
<tr>
<td></td>
<td>shipping groups separately.</td>
</tr>
<tr>
<td>INVALID_SHIP_SRVC</td>
<td>The selected service is not valid for international shipments. Please choose an international</td>
</tr>
<tr>
<td></td>
<td>service and retry your request.</td>
</tr>
<tr>
<td>INVALID_SHIP_TO_SATE</td>
<td>The Ship To State/Province Code is missing or invalid. Please enter the 2 to 5 character</td>
</tr>
<tr>
<td></td>
<td>abbreviation for the state or province of the address that contains it.</td>
</tr>
<tr>
<td>INVALID_SHIPPER_STATE</td>
<td>The Shipper State/Province Code is missing or invalid. Please enter the 2 to 5 character</td>
</tr>
<tr>
<td></td>
<td>abbreviation for the state or province of the address that contains it.</td>
</tr>
<tr>
<td>INVALID_SITE_CSTM_FILE</td>
<td>File is not a NetSuite site customization export file: it cannot be imported.</td>
</tr>
<tr>
<td>INVALID_SOAP_HEADER</td>
<td>Invalid SOAP Header: '{1}'. Value is '{2}'.</td>
</tr>
<tr>
<td>INVALID_SRCH</td>
<td>That search or mass update does not exist (internal id={1})</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>INVALID_SRCH_CRITERIA</td>
<td>The field rule value &quot;{1}&quot; is invalid for field type (2) with criterion &quot;(3).&quot;</td>
</tr>
<tr>
<td>INVALID_SRCH_CSTM_FLD</td>
<td>This search refers to custom field with id = (1) which either is restricted or is not applied to this record type.</td>
</tr>
<tr>
<td>INVALID_SRCH_FUNCTN</td>
<td>An nlobjSearchColumn contains an invalid function: {1}.</td>
</tr>
<tr>
<td>INVALID_SRCH_SORT</td>
<td>An nlobjSearchColumn that is not sortable contains a sort specification: {1}.</td>
</tr>
<tr>
<td>INVALID_SRCH_SUMMARY_TYP</td>
<td>An nlobjSearchFilter contains an invalid summary type: {1}.</td>
</tr>
<tr>
<td>INVALID_SRCH_TYP</td>
<td>Search Type not allowed as standalone search</td>
</tr>
<tr>
<td>INVALID_SRVC_ITEM_SUB</td>
<td>The service item '{1:service item}' is not available in the subsidiary of customer '{2:customer}'.</td>
</tr>
<tr>
<td>INVALID_SSO</td>
<td>Invalid SuiteSignOn reference: {1}. That SuiteSignOn object does not exist or has been marked as inactive.</td>
</tr>
<tr>
<td>INVALID_SSS_DEBUG_SESSION</td>
<td>You have exceeded the maximum allowable idle time for debugging scripts. To debug another script, simply reload the script debugger page and start a new debugging session.</td>
</tr>
<tr>
<td>INVALID_SSS_DEBUG_SESSION</td>
<td>You have canceled your current script debugging session.</td>
</tr>
<tr>
<td>INVALID_STATE</td>
<td>Merchant State Length Error. Provide a valid state.</td>
</tr>
<tr>
<td>INVALID_STATE</td>
<td>Signup prospect state '{1}' is invalid.</td>
</tr>
<tr>
<td>INVALID_STATUS</td>
<td>You may not change this issue's status from &quot;(1)&quot; to &quot;(2)&quot;.</td>
</tr>
<tr>
<td>INVALID_SUB</td>
<td>The subsidiary restrictions on this record are incompatible with those defined for account: {1}. Subsidiary access on this record must be a subset of those permitted by the account.</td>
</tr>
<tr>
<td>INVALID_SUB</td>
<td>The subsidiary restrictions on this record are incompatible with those defined for department: {1}. Subsidiary access on this record must be a subset of those permitted by the department.</td>
</tr>
<tr>
<td>INVALID_SUB</td>
<td>The subsidiary restrictions on this record are incompatible with those defined for item: {1}. Subsidiary access on this record must be a superset of those permitted by the item.</td>
</tr>
<tr>
<td>INVALID_SUB</td>
<td>The subsidiary restrictions on this record are incompatible with those defined for location: {1}. Subsidiary access on this record must be a subset of those permitted by the location.</td>
</tr>
<tr>
<td>INVALID_SUB</td>
<td>The Subsidiary selected doesn't match the bank account selected.</td>
</tr>
<tr>
<td>INVALID_SUB</td>
<td>This record does not support multiple subsidiary restrictions. You must choose a single subsidiary.</td>
</tr>
<tr>
<td>INVALID_SUB</td>
<td>Transaction references multiple subsidiaries</td>
</tr>
<tr>
<td>INVALID_SUB</td>
<td>You may not add members to a group/kit/assembly unless the subsidiaries for those members completely contain the subsidiaries of the group/kit/assembly.</td>
</tr>
<tr>
<td>INVALID_SUB</td>
<td>{1} can not be used with the selected subsidiary</td>
</tr>
<tr>
<td>INVALID_SUBLIST_DESC</td>
<td>Invalid sublist description - all sublists must appear exactly as they do in the WSDL (eg end with 'List')</td>
</tr>
<tr>
<td>INVALID_SUBSCRIPTION_STATUS</td>
<td>You cannot change the global subscription status from its current value of {1:status name}.</td>
</tr>
<tr>
<td>INVALID_SUBSCRIPTION_STATUS</td>
<td>You cannot set the global subscription status to the value {1:status name}.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>INVALID_SUMMARY_SRCH</td>
<td>In a summary search, you must sort by a result field with a summary function. Please go back and correct the sort by field on the results tab.</td>
</tr>
<tr>
<td>INVALID_SUPERVISOR</td>
<td>Employees cannot be their own supervisor.</td>
</tr>
<tr>
<td>INVALID_SUPERVISOR</td>
<td>You can’t insert this employee record as it would create a loop in the supervisor hierarchy.</td>
</tr>
<tr>
<td>INVALID_TASK_ID</td>
<td>The task ID: {1} is not valid. Please refer to the documentation for a list of supported task IDs.</td>
</tr>
<tr>
<td>INVALID_TASK_ID</td>
<td>You have specified an invalid task ID</td>
</tr>
<tr>
<td>INVALID_TASK_AMT</td>
<td>Invalid tax amount. Correct tax amount and retry request.</td>
</tr>
<tr>
<td>INVALID_TASK_CODE</td>
<td>Invalid Canadian Tax Code: {1}</td>
</tr>
<tr>
<td>INVALID_TASK_CODE_FOR_SUB</td>
<td>The selected tax code is not available in subsidiary.</td>
</tr>
<tr>
<td>INVALID_TASK_CODES</td>
<td>Invalid Tax Code(s): {1}</td>
</tr>
<tr>
<td>INVALID_TASK_PMT</td>
<td>You may not commit tax payment information prior to the start date of the Payroll Service.</td>
</tr>
<tr>
<td>INVALID_TASK_VALUE</td>
<td>GST and PST amount cannot be negative!</td>
</tr>
<tr>
<td>INVALID_TASK_VALUE</td>
<td>GST tax value is not a valid number: {1}</td>
</tr>
<tr>
<td>INVALID_TIME_FORMAT</td>
<td>{1} is not a valid time and it should use the following format h:mm a.</td>
</tr>
<tr>
<td>INVALID_TO_DATE</td>
<td>invalid 'to' date</td>
</tr>
<tr>
<td>INVALID_TRACKING_NUM</td>
<td>The tracking number is not valid.</td>
</tr>
<tr>
<td>INVALID_TRACKING_NUM</td>
<td>You have entered a tracking number that exceeds the maximum size of {1} characters: {2}. Multiple tracking numbers must be separated by spaces, tabs, or commas. Slash (/), semicolon (;), colon (:), or any other character that is not a space or a comma will be interpreted as a part of the tracking number.</td>
</tr>
<tr>
<td>INVALID_TRAN_ITEM_LINE</td>
<td>Item {1} can not be included on the {2} because it is not distributed as of the transaction date</td>
</tr>
<tr>
<td>INVALID_TRANS</td>
<td>This transaction is not valid.</td>
</tr>
<tr>
<td>INVALID_TRANS_COMPNT</td>
<td>You have entered an invalid component for this transaction.</td>
</tr>
<tr>
<td>INVALID_TRANS_ID</td>
<td>Invalid Transaction ID. Correct the transaction ID, then re-submit.</td>
</tr>
<tr>
<td>INVALID_TRANS_ID</td>
<td>Only sale transactions can be refunded. Provide a valid transaction ID.</td>
</tr>
<tr>
<td>INVALID_TRANS_SUB_ACCT</td>
<td>Transaction subsidiary {1} is not valid for account {2}. Please choose a different account.</td>
</tr>
<tr>
<td>INVALID_TRANS_SUB_CLASS</td>
<td>Transaction subsidiary {1} is not valid for class {2}. Please choose a different class.</td>
</tr>
<tr>
<td>INVALID_TRANS_SUB_DEPT</td>
<td>Transaction subsidiary {1} is not valid for department {2}. Please choose a different department.</td>
</tr>
<tr>
<td>INVALID_TRANS_SUB_ENTITY</td>
<td>Transaction subsidiary {1} is not valid for entity {2}. Please choose a different entity.</td>
</tr>
<tr>
<td>INVALID_TRANS_SUB_ITEM</td>
<td>Transaction subsidiary {1} is not valid for item {2}. Please choose a different item.</td>
</tr>
<tr>
<td>INVALID_TRANS_SUB_LOC</td>
<td>Transaction subsidiary {1} is not valid for location {2}. Please choose a different location.</td>
</tr>
<tr>
<td>INVALID_TRANS_TYP</td>
<td>Transaction type specified is incorrect.</td>
</tr>
<tr>
<td>INVALID_TRANSACTION_DATE</td>
<td>There are no Accounting Periods that cover this transaction date.</td>
</tr>
<tr>
<td>INVALID_TRANSACTION_DATE</td>
<td>Transaction date {1} is not valid. Transaction dates may be at most {2} years in the past and {3} years in the future.</td>
</tr>
</tbody>
</table>

SuiteTalk SOAP Web Services Platform Guide
<table>
<thead>
<tr>
<th>Error Code Returned</th>
<th>Long Description or Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_TRIAL_TYP</td>
<td>The trialtype is not available in the product specified.</td>
</tr>
<tr>
<td>INVALID_TYP</td>
<td>Invalid type (1), use (2)</td>
</tr>
<tr>
<td>INVALID_UNIT_TYP</td>
<td>On serialized items, you may not choose a units type that has fractional conversion rates.</td>
</tr>
<tr>
<td>INVALID_UNSUPRTD_RCRD_TYP</td>
<td>Invalid or unsupported record type: (1)</td>
</tr>
<tr>
<td>INVALID_UPS_ACCT</td>
<td>An invalid UPS Account Number was entered. Please verify you have entered the correct Shipper Number and re-submit the form.</td>
</tr>
<tr>
<td>INVALID_UPS_PACKG_WEIGHT</td>
<td>UPS requires a minimum package weight of .1 LBS and a maximum package weight of 150 LBS. Please adjust the package weights accordingly and resubmit the fulfillment.</td>
</tr>
<tr>
<td>INVALID_UPS_VALUES</td>
<td>UPS did not accept the entered values for the following fields. Please go back and correct these values:</td>
</tr>
<tr>
<td>INVALID_URL</td>
<td>Please begin the (1) url with &lt;b&gt;http://&lt;/b&gt; or &lt;b&gt;https://&lt;/b&gt; Examples of a valid (1) url are:&lt;b&gt;<a href="http://www.mydomain.com/image.gif">http://www.mydomain.com/image.gif</a>&lt;/b&gt; or &lt;b&gt;<a href="https://one.two.org/user-name/test.jpg">https://one.two.org/user-name/test.jpg</a>&lt;/b&gt;</td>
</tr>
<tr>
<td>INVALID_URL</td>
<td>Request for invalid URL: (1)</td>
</tr>
<tr>
<td>INVALID_URL_PARAM</td>
<td>Error: URL param (1)=&quot;(2)&quot; - expected an integer.</td>
</tr>
<tr>
<td>INVALID_VALUE</td>
<td>You have entered an invalid value (1) for (2).</td>
</tr>
<tr>
<td>INVALID_VAT_AMOUNT</td>
<td>VAT amount cannot be negative</td>
</tr>
<tr>
<td>INVALID_VAT_REGSTRTN_NUM</td>
<td>Invalid VAT registration number (1).</td>
</tr>
<tr>
<td>INVALID_VSOE_ALLOCTN</td>
<td>VSOE allocations must be greater than or equal to 0</td>
</tr>
<tr>
<td>INVALID_WEBSITE_SECTION</td>
<td>The Web site section you entered does not exist.</td>
</tr>
<tr>
<td>INVALID_WO</td>
<td>You have an invalid work order (1) or the order is already closed.</td>
</tr>
<tr>
<td>INVALID_WO_ITEM</td>
<td>Special Work Order Items can not be Drop Ship or Special Order</td>
</tr>
<tr>
<td>INVALID_WORLDPAY_ID</td>
<td>Exchange source does not recognize your WorldPay ID. Please check that it is correct.</td>
</tr>
<tr>
<td>INVALID_WS_VERSION</td>
<td>The async operation has been submitted through different endpoint version: (1).</td>
</tr>
<tr>
<td>INVALID_YEAR</td>
<td>Invalid year (1)</td>
</tr>
<tr>
<td>INVALID_YEAR_FORMAT</td>
<td>Illegal year format or value. Examples: 1999, 2000, 2001, etc.</td>
</tr>
<tr>
<td>INVALID_ZIP_CODE</td>
<td>Merchant Zip Length Error. Provide a valid 5 digit zip.</td>
</tr>
<tr>
<td>INVALID_ZIP_FILE</td>
<td>Invalid archive. Zip file must contain at least one file.</td>
</tr>
<tr>
<td>INVALID_ZIP_POST_CODE</td>
<td>The submitted Zip/Postal Code is invalid. This field may only contain a maximum of 16 digits, spaces, and the dash character (-).</td>
</tr>
<tr>
<td>INVENTORY_NUM_DISALLWD</td>
<td>Inventory numbers are only allowed on items with serial numbered or lot numbered items.</td>
</tr>
<tr>
<td>INVLAIAD_BOOLEAN_VALUE</td>
<td>You have entered an invalid boolean value. Please use true, false, T, or F for boolean values and resubmit your import.</td>
</tr>
<tr>
<td>IP_REQUEST</td>
<td>Your IP address (1) does not match any of the ipaddress rules specified for this account.</td>
</tr>
<tr>
<td>ISSUE_ASSIGNEE_DISALLWD</td>
<td>The specified assignee is disallowed for this issue's status.</td>
</tr>
<tr>
<td>ISSUE_PRODUCT_VERSION_MISMATCH</td>
<td>Cannot set issue (1) to (2) (3) and (4) (5) because that product is not associated with that version.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ISSUE_VERSION_BUILD_MISMATCH</td>
<td>Issue version and build do not match.</td>
</tr>
<tr>
<td>ITEM_ACCT_REQD</td>
<td>One of the items on this transaction has an amount but no account. Please fix the item and resubmit the transaction.</td>
</tr>
<tr>
<td>ITEM_ACCT_REQD</td>
<td>One of the items on this transaction has an amount but no account. Please fix the item and resubmit the transaction. It might be that you have recently elected to charge for shipping and have not assigned an account to the shipping item that is included in this transaction.</td>
</tr>
<tr>
<td>ITEM_ACCT_REQD</td>
<td>You must specify asset and COGS accounts for this inventory item.</td>
</tr>
<tr>
<td>ITEM_COUNT_MISMATCH</td>
<td>COGS_CORRECTION: 2 means of calculating the item count do not match for item: {1} vs {2})</td>
</tr>
<tr>
<td>ITEM_COUNT_MISMATCH</td>
<td>COGS_CORRECTION: 2 means of calculating the item count do not match for item: {1} vs {2}) There are transactions in the system in which this item is used but the asset account for that item is not the current Asset Account in the item record</td>
</tr>
<tr>
<td>ITEM_IS_UNAVAILABLE</td>
<td>(Item is unavailable)</td>
</tr>
<tr>
<td>ITEM_NAME_MUST_BE_UNIQUE</td>
<td>An item with that name already exists. Please choose another name</td>
</tr>
<tr>
<td>ITEM_NOT_UNIQUE</td>
<td>The item [{1}] is not unique.</td>
</tr>
<tr>
<td>ITEM_PARAM_REQD_IN_URL</td>
<td>Error - Item parameter (id=nnn) was not provided on the URL</td>
</tr>
<tr>
<td>ITEM_QTY_AMT_MISMATCH</td>
<td>Inventory and assembly items cannot have zero quantity and non-zero amount.</td>
</tr>
<tr>
<td>ITEM_TYP_REQS_UNIT</td>
<td>Items of type {1} require {2} unit</td>
</tr>
<tr>
<td>ITEM_TYP_REQS_UNIT</td>
<td>Items of type (1) subtype (2) require (3) unit</td>
</tr>
<tr>
<td>JE_AMOUNTS_MUST_BALANCE</td>
<td>The amounts in the journal entry must balance.</td>
</tr>
<tr>
<td>JE_LINE_MISSING_REQD_DATA</td>
<td>{1} are mandatory on all lines of the journal entry.</td>
</tr>
<tr>
<td>JE_MAX.ONE_LINE</td>
<td>Journal Entries can have a maximum of (1) lines.</td>
</tr>
<tr>
<td>JE_REV_REC_IN_PROGRESS</td>
<td>This account is currently processing Revenue Recognition Journal Entries. Only one such process is allowed at a time.</td>
</tr>
<tr>
<td>JE_UNEXPECTED_ERROR</td>
<td>Journal Entries failed to be created due to unexpected error.</td>
</tr>
<tr>
<td>JOB_NOT_COMPLETE</td>
<td>The specified job is not complete yet</td>
</tr>
<tr>
<td>JS_EXCEPTION</td>
<td>A JavaScript Exception was thrown</td>
</tr>
<tr>
<td>KEY_REQD</td>
<td>Empty key not allowed for (1)</td>
</tr>
<tr>
<td>KPI_SETUP_REQD</td>
<td>Please enable the Forecast (1)) &amp; Quota KPIs</td>
</tr>
<tr>
<td>KPI_SETUP_REQD</td>
<td>Please enable the Forecast &amp; Quota KPIs</td>
</tr>
<tr>
<td>KPI_SETUP_REQD</td>
<td>please enable the Sales &amp; Forecast KPIs</td>
</tr>
<tr>
<td>KPI_SETUP_REQD</td>
<td>Please enable the Sales &amp; Forecast (1) KPIs</td>
</tr>
<tr>
<td>KPI_SETUP_REQD</td>
<td>Please enable the Sales &amp; Quota KPIs</td>
</tr>
<tr>
<td>LABEL_REQD</td>
<td>Please enter a value for Label</td>
</tr>
<tr>
<td>LANGUAGE_SETUP_REQD</td>
<td>Please go to company preference to add language to translate.</td>
</tr>
<tr>
<td>LINK_LINES_TO_ONE_ORD</td>
<td>Lines on this record can only be linked to a single order.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>LINKED_ACCT_DONT_MATCH</td>
<td>You are attempting to link transaction line items, but items on the lines do not match. This can happen when you create a fulfillment from a sales order, a receipt from a purchase order, an invoice from a sales order, a vendor bill from a purchase order, or a reimbursement from a purchase. Please verify that items in the transaction you are creating match the items in the originating transaction.</td>
</tr>
<tr>
<td>LINKED_ENTITIES_DONT_MATCH</td>
<td>You are attempting to link transactions, but the entities on the transactions do not match.</td>
</tr>
<tr>
<td>LINKED_ITEMS_DONT_MATCH</td>
<td>Linked items don't match</td>
</tr>
<tr>
<td>LIST_ID_REQD</td>
<td>Required field missing in a related list. You must set {1}.</td>
</tr>
<tr>
<td>LIST_KEY_REQD</td>
<td>There is no list key for field {1} of list {2}. Please assign a key and resubmit your task.</td>
</tr>
<tr>
<td>LOCATIONS_IN_USE</td>
<td>Your classes cannot be converted to locations because your existing location records are referred to by transactions or other records. These location records cannot be overwritten.</td>
</tr>
<tr>
<td>LOCATIONS_SETUP_REQD</td>
<td>You must first define locations (Lists-&gt;Locations-&gt;New) before you can distribute inventory.</td>
</tr>
<tr>
<td>LOCATIONS_SETUP_REQD</td>
<td>You must first define locations (Lists-&gt;Locations-&gt;New) before you can transfer inventory.</td>
</tr>
<tr>
<td>LOCATION_REQD</td>
<td>You must specify a location to use {1} numbers when Multi-Location Inventory is enabled</td>
</tr>
<tr>
<td>LOCKED_DASHBOARD</td>
<td>Your dashboard has been set up and locked by an administrator. Please contact them for details.</td>
</tr>
<tr>
<td>LOGIN_DISABLED</td>
<td>Invalid login. Customer access is disabled.</td>
</tr>
<tr>
<td>LOGIN_DISABLED</td>
<td>Login access has been disabled for this role.</td>
</tr>
<tr>
<td>LOGIN_DISABLED</td>
<td>Your access to {1} has been deactivated. Please contact the company's administrator to reactivate your access.</td>
</tr>
<tr>
<td>LOGIN_DISABLED</td>
<td>Your access to this account has been removed or disabled. Please contact the account administrator.</td>
</tr>
<tr>
<td>LOGIN_DISABLED_PARTNER_CTR</td>
<td>Disabled login: Advanced Partner Center access has been disabled by the account administrator.</td>
</tr>
<tr>
<td>LOGIN_DISABLED_PARTNER_CTR</td>
<td>Disabled login: Standard Partner Center access has been disabled by the account administrator.</td>
</tr>
<tr>
<td>LOGIN_EMAIL_REQD</td>
<td>Invalid login. You must provide an email address.</td>
</tr>
<tr>
<td>LOGIN_NAME_AND_PSWD_REQD</td>
<td>Please enter both a user name and a password.</td>
</tr>
<tr>
<td>LOGIN_REQD</td>
<td>You must log in before accessing this page.</td>
</tr>
<tr>
<td>LOST_UPSELL_CRITERIA</td>
<td>Your upsell criteria were lost. This is probably due to a transient condition such as a server reboot. Please click here to go back and try again.</td>
</tr>
<tr>
<td>MACHN_LIST_KEY_NAMES_REQD</td>
<td>Server application error: no list key names are defined for field {1} of record of type {2}.</td>
</tr>
<tr>
<td>MANDATORY_PRD_TYPE_REQD</td>
<td>Please select the mandatory period type.</td>
</tr>
<tr>
<td>MASS_UPDATE_CRITERIA_LOST</td>
<td>Your mass update criteria were lost. This is probably due to a transient condition such as a server reboot. Please click here to go back and try again.</td>
</tr>
<tr>
<td>MATCHING_CUR_SUB_REQD</td>
<td>The parent specified must have the same currency and subsidiary as the child.</td>
</tr>
<tr>
<td>MATCHING_SERIAL_NUM_REQD</td>
<td>The serial numbers on a transfer order receipt must have been fulfilled.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MATRIX_INFO_TEMP_LOST</td>
<td>Matrix item information was lost. This was probably due to a transient condition like a server reboot. Please try again.</td>
</tr>
<tr>
<td>MATRIX_SUBITEM_NAME_TOO_LONG</td>
<td>The following matrix sub-item name is too long (80 character max):&lt;p&gt; (1) &lt;p&gt; Please shorten your parent item name or your option abbreviations.</td>
</tr>
<tr>
<td>MAX_16_LINES_ALLWD_PER_BILLPAY</td>
<td>A maximum of 16 lines per payee can be applied per online bill payment.</td>
</tr>
<tr>
<td>MAX_200_LINES_ALLWD_ON_TRANS</td>
<td>Journal Entries can have a maximum of 200 lines.</td>
</tr>
<tr>
<td>MAX_BARCODE_PRINT_EXCEEDED</td>
<td>A maximum of {1} barcodes can be printed at a time.</td>
</tr>
<tr>
<td>MAX_BULK_MERGE_RCRDS_EXCEEDED</td>
<td>You cannot perform a bulk merge operation with a group larger than {1} records</td>
</tr>
<tr>
<td>MAX_EMAILS_EXCEEDED</td>
<td>This campaign email event exceeds the number of emails ({1}) that can be sent per month without setting up a default campaign domain or specifying one on the campaign email template.</td>
</tr>
<tr>
<td>MAX_EMAILS_EXCEEDED</td>
<td>This merge operation exceeds the number of emails ({1}) that can be sent per month without setting up a bulk merge domain or specifying one on the email template.</td>
</tr>
<tr>
<td>MAX_RCRDS_EXCEEDED</td>
<td>The maximum number ({1}) of records allowed for a (2) operation has been exceeded.</td>
</tr>
<tr>
<td>MAX_VALUES_EXCEEDED</td>
<td>Too many values specified for a MultiSelectField, the maximum is 1000</td>
</tr>
<tr>
<td>MEDIA_FILE_INVALID_JSCRIPT</td>
<td>Media file was of type javascript and would not compile. Error on line:</td>
</tr>
<tr>
<td>MEDIA_NOT_FOUND</td>
<td>Media item not found {1}</td>
</tr>
<tr>
<td>MEDIA_NOT_INITIALIZED</td>
<td>Media Item cannot be initialized</td>
</tr>
<tr>
<td>MEMORIZED_TRANS_ERROR</td>
<td>failed retrieving ‘chargeit’ record from trancard while processing memorized tran</td>
</tr>
<tr>
<td>MEMORIZED_TRANS_ERROR</td>
<td>A failure occurred while trying to enter one of your automated memorized transactions. The reason for this failure is shown in the description field on this page. Because of the failure, the system has changed this memorized transaction from an automated one to a reminder. You may click the link above to view the memorized transaction in question and make any needed changes to it. When you are done with the changes, click Memorize and then Cancel. This will modify your existing memorized transaction rather than creating a new one. When filling out the Memorized Transaction Form, you may choose to select ‘Automatic’ to resume automated posting of this transaction.</td>
</tr>
<tr>
<td>MERGE_OPERATION_DISALLWD</td>
<td>You cannot perform merge operations on records that belong to your group.</td>
</tr>
<tr>
<td>MERGE_RCRD_REQD</td>
<td>You must specify a record to merge into</td>
</tr>
<tr>
<td>METAVANTE_ERROR</td>
<td>This Online Bill Payment has been {1} in your NetSuite account. However, the payment was not able to be {2} at Metavante. &lt;BR&gt;To (3) this payment at Metavante, please contact NetSuite Customer Support immediately, using the options provided in the Support &gt; Customer Service section of your NetSuite account.</td>
</tr>
<tr>
<td>METAVANTE_SECRET_ANSWER_REQD</td>
<td>Missing Secret Answer. A secret answer is required by the Metavante CSP service. It cannot be null or empty.</td>
</tr>
<tr>
<td>METAVANTE_SECRET_QUESTION_REQD</td>
<td>Missing or invalid Secret Question ID. A valid Secret Question is required by the Metavante CSP service. Please refer to Table 2, above, for a list of valid Secret Question IDs.</td>
</tr>
<tr>
<td>METAVANTE_SETUP_REQD</td>
<td>Your NetSuite account is not currently integrated with a Metavante Online Bill Pay account.&lt;br&gt;To set up an active account, you need to reapply to Metavante.&lt;br&gt;Go to</td>
</tr>
</tbody>
</table>
### Error Status Codes

<table>
<thead>
<tr>
<th>Error Code Returned</th>
<th>Long Description or Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>METAVANTE_TEMP_UNAVAILBL</td>
<td>Metavante is temporarily unavailable. Please try again later. If you would like to print the payment to mail yourself, click Back, and then click the date of the payment on the Approve Online Bill Payments page. When the payment's detail record appears, clear the Bill Pay box and either check the To Be Printed box and click Submit or click the Print button.</td>
</tr>
<tr>
<td>MISMATCH_EVENT_ISSUE_STATUS</td>
<td>Event status (1) and issue base status (2) do not match</td>
</tr>
<tr>
<td>MISMATCH_ISSUE_PRODUCT_VERSION</td>
<td>Issue product and version do not match.</td>
</tr>
<tr>
<td>MISMATCH_SALES_CONTRIBUTION</td>
<td>Sales team sales rep total does not equal 100%, (1) sales reps, (2) total contribution.</td>
</tr>
<tr>
<td>MISMATCHED_CURRENCY</td>
<td>The transaction currency does not match the names currency</td>
</tr>
<tr>
<td>MISMATCHED_QTY_PRICING</td>
<td>Quantities do not match across pricings</td>
</tr>
<tr>
<td>MISMATCHED_SEARCH_PARENTHESIS</td>
<td>Search error: Parentheses are unbalanced.</td>
</tr>
<tr>
<td>MISSING_ACCT_PRD</td>
<td>You are attempting to create an amortization or revenue recognition schedule outside the range of available accounting periods. Please adjust the periods on this transaction or go to Setup&gt;Accounting&gt;Manage Accounting Periods to set up more periods.</td>
</tr>
<tr>
<td>MISSING_CRNCY_EXCH_RATE</td>
<td>No currency conversion rate defined</td>
</tr>
<tr>
<td>MISSING_ENUM</td>
<td>No Enumerated Value (1) for Enumerated Type (2)</td>
</tr>
<tr>
<td>MISSING_REQD_FLD</td>
<td>Missing required value for mandatory field: (1)</td>
</tr>
<tr>
<td>MISSING_REQD_FLD</td>
<td>The Company record does not have all required fields set. Please ensure the State, Zip/Postal Code, and Country fields are set and try your request again.</td>
</tr>
<tr>
<td>MISSNG_ACCT_PRD</td>
<td>Unable to find an Accounting Period for the allocation date.</td>
</tr>
<tr>
<td>MISSNG_REV_REC_RCRD</td>
<td>Unable to locate Revenue Recognition records.</td>
</tr>
<tr>
<td>MISSNG_SO_REV_REC_PARAMS</td>
<td>Unable to get Revenue Recognition parameters from originating sales order.</td>
</tr>
<tr>
<td>MISSNG_SO_START_END_DATES</td>
<td>Unable to acquire start and end date from Sales Order.</td>
</tr>
<tr>
<td>MLI_REQD</td>
<td>Multi-location Inventory Error (MLI_LOCATION_REQUIRED): this transaction or its items must have locations.</td>
</tr>
<tr>
<td>MLTIPLE_TAX_LINES_DISALLWD</td>
<td>Multiple Tax lines for line item in transaction:</td>
</tr>
<tr>
<td>MSGN_FIELD_OWRTE_MUST_BE_TRUE</td>
<td>The missingFieldOverwrite attribute must be true when updating a salesOrder.</td>
</tr>
<tr>
<td>MST_UPDATE_ITEMS_THEN_RATES</td>
<td>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.</td>
</tr>
<tr>
<td>MULTI_ACCT_CANT_CHANGE_PSWD</td>
<td>The password cannot be changed here because the email address is associated with multiple accounts. The user must change their password via the link in the settings portal of the home page.</td>
</tr>
</tbody>
</table>
| MULTI_LOC_INVNT_ERROR | Multi-Location Inventory Error: You may not create an Assembly Build transaction with an assembly item that has not been distributed and member items that have been distributed. You must create an Inventory Distribution transaction for the assembly item before building it. You also may not create an Assembly Build transaction on a date prior to the
<table>
<thead>
<tr>
<th>Error Code Returned</th>
<th>Long Description or Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>409</td>
<td>Error Code Returned</td>
</tr>
<tr>
<td>409</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>DISTRIBUTION_DATE_OF_THE.Assembly</td>
<td>distribution date of the assembly but after the distribution date of any of the member items.</td>
</tr>
<tr>
<td>MULTI_PRIMARY_PARTNER.DISALLWD</td>
<td>You are not allowed to select multiple primary partners.</td>
</tr>
<tr>
<td>MULTI_SHIP ROUTES REQD</td>
<td>{1} {2} has multiple shipping routes enabled, which is only supported in version 2008_2 and newer. You are not allowed to update any shipping fields on this record.</td>
</tr>
<tr>
<td>MULTISELECT_TYPE REQD</td>
<td>Server application error: no multiselect type is defined for field (1) of (2) record type.</td>
</tr>
<tr>
<td>MUST_DEFINE_BASE_UNIT</td>
<td>One unit must be designated as the base unit.</td>
</tr>
<tr>
<td>MUST_RESUBMIT_RCRD</td>
<td>Configuration changes have been made to your account. You must resubmit your record.</td>
</tr>
<tr>
<td>NAME_ALREADY_IN_USE</td>
<td>A mass update has already been saved with that name. Please use a different name.</td>
</tr>
<tr>
<td>NAME_ALREADY_IN_USE</td>
<td>A search has already been saved with that name. Please use a different name.</td>
</tr>
<tr>
<td>NAME_REQD</td>
<td>Missing Name. Name is a required field and it cannot be null or empty.</td>
</tr>
<tr>
<td>NAME_TYPE_FLDR_FIELDS_REQD</td>
<td>missing required fields : name, type, and folder</td>
</tr>
<tr>
<td>NARROW_KEYWORD_SEARCH</td>
<td>Please provide more detailed keywords so your search does not return too many results.</td>
</tr>
<tr>
<td>NEED_BILL_VARIANCE_ACCT</td>
<td>Bill variance account is missing</td>
</tr>
<tr>
<td>NEGATIVE_PAYMENT_DISALLWD</td>
<td>Negative payments not allowed</td>
</tr>
<tr>
<td>NEGATIVE_TAX_RATE_DISALLWD</td>
<td>A Tax rate cannot be negative</td>
</tr>
<tr>
<td>NEW_CONNECTION_DISALLWD</td>
<td>Not allowed to create new connections.</td>
</tr>
<tr>
<td>NEXUS_REQD</td>
<td>No tax agency defined for subsidiary: subsidiary (1) is not linked to nexus (2)</td>
</tr>
<tr>
<td>NO_DATA_FOUND</td>
<td>No data was found</td>
</tr>
<tr>
<td>NO_EXPENSES_FOR_PRD</td>
<td>The Allocation sources or destinations did not have any expenses associated with them for the selected period.</td>
</tr>
<tr>
<td>NO_ITEMS_TO_PRINT</td>
<td>There are no items to print</td>
</tr>
<tr>
<td>NO_MASS_UPDATES_RUNNING</td>
<td>There are currently no mass updates running.</td>
</tr>
<tr>
<td>NO_MTRX_ITEMS_TO_UPDATE</td>
<td>There are no matrix subitems to update.</td>
</tr>
<tr>
<td>NO_ORD_SHPMNT</td>
<td>There is no shipment on that order.</td>
</tr>
<tr>
<td>NO_RCRD_FOR_USER</td>
<td>There is no record for this user in the company's entity table. (emaillogin.semail='{1}', kentity={2})</td>
</tr>
<tr>
<td>NO_RCRDS_MATCH</td>
<td>No Records matched your request.</td>
</tr>
<tr>
<td>NO_SCHEDUL_APPLIED</td>
<td>There were no schedules that need to applied to the particular period.</td>
</tr>
<tr>
<td>NO_SCHEDUL_APPLIED</td>
<td>There were no schedules that need to be applied to the input accounting period.</td>
</tr>
<tr>
<td>NON_ADMIN_CANNOT_INITIATE_LINK</td>
<td>This user cannot integrate with a partner.</td>
</tr>
<tr>
<td>NONMATCHING_EMAILS</td>
<td>Email addresses don't match</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>NONUNIQUE_INDEX_VALUE</td>
<td>An attempt was made to insert a row with a non-unique index value. Dynamic SQL being executed []</td>
</tr>
<tr>
<td>NONZERO_AMT_REQD</td>
<td>You did not enter non-zero amounts for any accounts.</td>
</tr>
<tr>
<td>NONZERO_QTY_REQD</td>
<td>Build quantity must be greater than zero.</td>
</tr>
<tr>
<td>NONZERO_WEIGHT_REQD</td>
<td>Selected service must have a weight greater than zero.</td>
</tr>
<tr>
<td>NOT_AN_INVITEE</td>
<td>You are not on the invitee list for event.</td>
</tr>
<tr>
<td>NOT_IN_INVT</td>
<td>You may not distribute [1] numbers that are not currently in inventory. You attempted to distribute the following [1] numbers that were not in inventory: [2]</td>
</tr>
<tr>
<td>NULL_CHECK_NUMBER</td>
<td>Null Check Number</td>
</tr>
<tr>
<td>NUM_ITEMS_GRTR_THAN_QTY</td>
<td>The number of [1] entered ([2]) is greater than the item quantity ([3])</td>
</tr>
<tr>
<td>NUM_ITEMS_NOT_EQUAL_TO_QTY</td>
<td>The number of [1] entered ([2]) is not equal to the item quantity ([3])</td>
</tr>
<tr>
<td>NUM_REQD_FOR_FIRST_LABEL</td>
<td>No number was specified for the first label.</td>
</tr>
<tr>
<td>NUMERIC_CHECK_NUM_REQD</td>
<td>Invalid Check Number. Check number must be a numeric value and can be at most 7 digits long.</td>
</tr>
<tr>
<td>NUMERIC_REF_NUM_REQD</td>
<td>Reference Number Must Be Numeric. Please provide a valid number and re-submit.</td>
</tr>
<tr>
<td>O1_FEATURE_REQD</td>
<td>You have not enabled Outlook Integration feature for your account.</td>
</tr>
<tr>
<td>O1_PERMISSION_REQD</td>
<td>You do not have permission to access Outlook Integration feature.</td>
</tr>
<tr>
<td>ONE_ADMIN_REQD_PER_ACCT</td>
<td>This operation would leave your account without an active Administrator. To successfully perform the mass update, please deselect at least one entity with an Administrator role.</td>
</tr>
<tr>
<td>ONE_ADMIN_REQD_PER_ACCT</td>
<td>You can't delete this employee. No administrators for this account would remain.</td>
</tr>
<tr>
<td>ONE_ADMIN_REQD_PER_ACCT</td>
<td>You can't inactivate [1]. The account would be left with no active administrators.</td>
</tr>
<tr>
<td>ONE_ADMIN_REQD_PER_ACCT</td>
<td>You can't remove the administrator role from this user. No administrators for this account would remain.</td>
</tr>
<tr>
<td>ONE_EMPL_REQD</td>
<td>At least one employee is required to process payroll</td>
</tr>
<tr>
<td>ONE_PAY_ITEM_PER_EMPL</td>
<td>An employee may not have multiple instances of payroll items. Go &lt;a href=&quot;javascript:history.go(-1);&quot;&gt;back&lt;/a&gt;, remove these items and resubmit.</td>
</tr>
<tr>
<td>ONE_POSITIVE_VALUE_REQD</td>
<td>You must enter at least one positive value for at least one item.</td>
</tr>
<tr>
<td>ONE_RCRD_REQD_FOR_MASS_UPDATE</td>
<td>Please create at least one [1] before using this mass update.</td>
</tr>
<tr>
<td>ONE_ROLE_REQD</td>
<td>You can't inactivate all roles. You would not be able to log in.</td>
</tr>
<tr>
<td>ONLINE_BANK_FILE_REQD</td>
<td>You must first upload an Online Bank file before using the Online Bank Statement.</td>
</tr>
<tr>
<td>ONLINE_BILL_PAY_DUP</td>
<td>Another entity with the same online bill pay name, account, and address already exists. Please find and use that entity for online bill pay.</td>
</tr>
<tr>
<td>ONLINE_BILL_PAY_SETUP_REQD</td>
<td>&lt;b&gt;{1}&lt;/b&gt; is not set up for Online Bill Pay. To set up this payee, click Go Back. When the Approve Online Bill Payments page appears, click Enable Payee in the Enabled for Billpay column. &lt;br&gt;When the payee's record appears, check Enable Online Bill Pay and submit these required fields:&lt;ul&gt;&lt;li&gt;Legal Name&lt;/li&gt;&lt;li&gt;Print on Check As&lt;/li&gt;&lt;li&gt;Phone&lt;/li&gt;&lt;li&gt;Billing Address&lt;/li&gt;&lt;/ul&gt;&lt;br&gt;Then, go to Transactions &gt; Approve Online Bill Payments to approve the payment.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ONLINE_FORM_DSNT_EXIST</td>
<td>This online form does not exist.</td>
</tr>
<tr>
<td>ONLINE_FORM_EMPTY</td>
<td>The online form you requested is empty.</td>
</tr>
<tr>
<td>ONLINE_FORM_ID_REQD</td>
<td>Missing required online form ID</td>
</tr>
<tr>
<td>ONLINE_FORM_USER_ACCESS_ONLY</td>
<td>This form is only accessible to online form users.</td>
</tr>
<tr>
<td>ONLINE_ORD_FEATURE_DISABLED</td>
<td>Can't open store for (1). This company does not have the <code>&lt;b&gt;Use Sales Orders&lt;/b&gt;</code> feature enabled. The feature is required for customers to make online purchases.</td>
</tr>
<tr>
<td>ONLY_ONE_CONTRIB_ITEM_REQD</td>
<td>Only one instance of a company contribution item is allowed on an employee record.</td>
</tr>
<tr>
<td>ONLY_ONE_DEDCT_ITEM_REQD</td>
<td>Only one instance of a deduction item is allowed on an employee record.</td>
</tr>
<tr>
<td>ONLY_ONE_DISTRIB_ALLWD</td>
<td>You may not distribute (1) numbers more than one time. You attempted to distribute the following (1) numbers more than one time: (2)</td>
</tr>
<tr>
<td>ONLY_ONE_EARNING_ITEM_REQD</td>
<td>Only one instance of an earning item is allowed on an employee record.</td>
</tr>
<tr>
<td>ONLY_ONE_LOT_NUM_ALLWD</td>
<td>You may not enter more than a single serial/lot number before an item is selected.</td>
</tr>
<tr>
<td>ONLY_ONE_PREF_BIN_ALLWD</td>
<td>There may be at most one preferred bin per location for an item. The following location has more than one preferred bin for this item: (1)</td>
</tr>
<tr>
<td>ONLY_ONE_PREF_BIN_ALLWD</td>
<td>You may not have more than one preferred bin per item.</td>
</tr>
<tr>
<td>ONLY_ONE_UNIT_AS_BASE_UNIT</td>
<td>Only one unit may be designated as the base unit.</td>
</tr>
<tr>
<td>ONLY_ONE_UPLOAD_ALLWD</td>
<td>You cannot upload more than one file at a time.</td>
</tr>
<tr>
<td>ONLY_ONE_WITHLD_ITEM_REQD</td>
<td>Only one instance of a withholding item is allowed on an employee record.</td>
</tr>
<tr>
<td>OPENID_DOMAIN_IN_USE</td>
<td>This domain is already in use by another NetSuite account, and a domain cannot be used by multiple accounts.</td>
</tr>
<tr>
<td>OPENID_NOT_ENABLED</td>
<td>The OpenID Single Sign-on feature is not enabled in this account.</td>
</tr>
<tr>
<td>OPRTN_UNAVAILBL_TO_GATEWAY</td>
<td></td>
</tr>
<tr>
<td>ORD_ALREADY_APPRVD</td>
<td>You cannot cancel this order because it has already been approved.</td>
</tr>
<tr>
<td>ORD_UNAVAILBL_TO_FULFILL</td>
<td>This order is not available for fulfillment. The current Google Order Financial State is (1)</td>
</tr>
<tr>
<td>ORDER_DSNT_EXIST</td>
<td>That order does not exist.</td>
</tr>
<tr>
<td>OTHER_PMT_AUTH_IN_PROGRESS</td>
<td>Another payment authorization is currently in progress for this order. Please try again in a few minutes.</td>
</tr>
<tr>
<td>OVER_FULFILL_DISALLWD</td>
<td>You cannot over-fulfill an item unless you have selected the 'Allow Overage on Item Fulfillments' preference.</td>
</tr>
<tr>
<td>OVER_FULFILL_RECEIV_DISALLWD</td>
<td>Transfer orders cannot be overfulfilled or overreceived</td>
</tr>
<tr>
<td>OVERAGE_DISALLWD</td>
<td>Overage is not allowed.</td>
</tr>
<tr>
<td>OVERLAPPING_PRDS_DISALLWD</td>
<td>Illegal period structure. Overlapping periods.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OVERLAPPING_PRDS_DISALLOWED</td>
<td>There is an overlapping period. Please check your Active or Inactive Periods to ensure that there is not an existing period.</td>
</tr>
<tr>
<td>OWNER_REQD</td>
<td>You cannot make a contact private without an owner</td>
</tr>
<tr>
<td>PACKAGE_WEIGHT_REQD</td>
<td>Attempted to create a package without specifying a nonzero package weight.</td>
</tr>
<tr>
<td>PKG_LVL_REF_DISALLOWED</td>
<td>Package level reference numbers are not allowed for shipments whose origin/destination pair is not US/US or Puerto Rico/Puerto Rico.</td>
</tr>
<tr>
<td>PKG_VALUE_TOO_LARGE</td>
<td>Package declared value cannot be greater than $999.00 USD</td>
</tr>
<tr>
<td>PARENT_CANT_ITSELF_BE_MEMBRER</td>
<td>Parent item can not be a member of itself</td>
</tr>
<tr>
<td>PARENT_MUST_BE_MATRIX_ITEM</td>
<td>A Child matrix item's parent must be a matrix item</td>
</tr>
<tr>
<td>PARENT_REQD</td>
<td>A Child matrix item must have its parent specified</td>
</tr>
<tr>
<td>PARSING_ERROR</td>
<td>Unable to parse value into the correct type. Problem occured on record type: (1), internalid: (2), field: (3), value: (4)</td>
</tr>
<tr>
<td>PARTIAL_FULFILL_RECEIvable_DISALLOWED</td>
<td>Transfer orders can not be partially fulfilled or partially received</td>
</tr>
<tr>
<td>PARTNER_ACCESS_DENIED</td>
<td>Partners do not have access to this item.</td>
</tr>
<tr>
<td>PARTNER_ACCT_NOT_LINKED</td>
<td>Account ({1}) from partner ({2}) is not linked.</td>
</tr>
<tr>
<td>PARTNER_CODE_ALREADY_USED</td>
<td>A partner with that partner code ({1}) already exists.</td>
</tr>
<tr>
<td>PAY_HOLD_ON_SO</td>
<td>This sales order cannot be fulfilled because it has a payment hold.</td>
</tr>
<tr>
<td>PAYCHECK_ALREADY_PAID</td>
<td>You are trying to edit a paycheck that is already paid by direct deposit. A paycheck cannot be edited after funds have been processed by the Automated Clearing House (ACH).</td>
</tr>
<tr>
<td>PAYCHECK_IN_USE</td>
<td>You cannot clear this paycheck because it is linked to by one or more liability payments. You must delete or void those transactions first</td>
</tr>
<tr>
<td>PAYEE_REQD_FOR_PMT</td>
<td>Your payment has been recorded, but an online bill pay payment will not be made because no payee was specified. You should return to the payment screen if you wish to print the check.</td>
</tr>
<tr>
<td>PAYPAL_FUND_SOURCE_REQD</td>
<td>Please return to PayPal to select a different funding source.</td>
</tr>
<tr>
<td>PAYPAL_INVALID_PMT_METHOD</td>
<td>PayPal is unable to process this payment. Please select an alternate payment method.</td>
</tr>
<tr>
<td>PAYPAL_INVALID_PMT_METHOD</td>
<td>Your PayPal account is not configured to use Express Checkout. Please follow directions on the PayPal payment method record.</td>
</tr>
<tr>
<td>PAYPAL_PMNT_NOTIFICATION</td>
<td>PayPal Payment Notification</td>
</tr>
<tr>
<td>PAYPAL_SETUP_REQD</td>
<td>The account referenced by this paypal id is not setup to use express checkout. Please return to the paypal setup page and follow directions for setting up paypal express checkout.</td>
</tr>
<tr>
<td>PAYROLL_COMMITTED</td>
<td>You are trying to edit a paycheck reversal that is in a committed Payroll Batch.</td>
</tr>
<tr>
<td>PAYROLL_COMMITTED</td>
<td>You are trying to edit a paycheck that is in a committed Payroll Batch.</td>
</tr>
<tr>
<td>PAYROLL_COMMITTED</td>
<td>You are trying to modify a committed payroll batch or a document on a committed payroll batch.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PAYROLL_EXPENSE_ACCT_REQD</td>
<td>Please select an expense account for payroll item &lt;a href=&quot;/app/common/item/payrollitem.nl?id={1}&amp;e=T&quot;&gt;{2}&lt;/a&gt;</td>
</tr>
<tr>
<td>PAYROLL_EXPENSE_ACCT_REQD</td>
<td>Please select a expense account for payroll item &lt;a href=&quot;/app/common/item/payrollitem.nl?id={1}&amp;e=T&quot;&gt;{2}&lt;/a&gt;</td>
</tr>
<tr>
<td>PAYROLL_FEATURE_DISABLED</td>
<td>You have not enabled the Payroll feature.</td>
</tr>
<tr>
<td>PAYROLL_FEATURE_UNAVAILABLE</td>
<td>You are trying to edit a Pay Cheque - Payroll is not available in NetSuite Canada.</td>
</tr>
<tr>
<td>PAYROLL_IN_PROCESS</td>
<td>A payroll process is currently running in this account. Please try again in a few minutes.</td>
</tr>
<tr>
<td>PAYROLL_ITEM_DELETE_DISALLWD</td>
<td>Unable to remove payroll item: (1) - There are existing transactions for this payroll item.</td>
</tr>
<tr>
<td></td>
<td>You may mark it inactive instead.</td>
</tr>
<tr>
<td>PAYROLL_LIABILITY_ACCT_REQD</td>
<td>Please select a liability account for payroll item &lt;a href=&quot;/app/common/item/payrollitem.nl?id={1}&amp;e=T&quot;&gt;{2}&lt;/a&gt;</td>
</tr>
<tr>
<td>PAYROLL_MAINTENANCE</td>
<td>The Payroll feature in your account is undergoing routine maintenance from (1) Pacific Time on (2) to (3) Pacific Time on (4). We apologize for any inconvenience this causes you.</td>
</tr>
<tr>
<td>PAYROLL_SETUP_REQD</td>
<td>The default payroll expense account is missing.&lt;br&gt;Please go to Setup &gt; Payroll &gt; Set Up Payroll and set the default payroll expense account.</td>
</tr>
<tr>
<td>PAYROLL_SETUP_REQD</td>
<td>You are trying to create a paycheck before your payroll setup is complete. Please complete your payroll setup.</td>
</tr>
<tr>
<td>PAYROLL_SETUP_REQD</td>
<td>You are trying to process payroll before your payroll setup is complete. Please complete your payroll setup.</td>
</tr>
<tr>
<td>PAYROLL_UPDATE_REQD</td>
<td>You made changes that require you to update your payroll information. Click &lt;a href=&quot;/app/payroll/managepayroll.nl&quot;&gt;here&lt;/a&gt; to commit updates to the payroll service.</td>
</tr>
<tr>
<td>PERMISSION_VIOLATION</td>
<td>Permission Violation: you may not access this record.</td>
</tr>
<tr>
<td>PERMISSION_VIOLATION</td>
<td>Permission Violation: you may no longer edit this record.</td>
</tr>
<tr>
<td>PHONE_NUM_REQD</td>
<td>Please provide a phone number.</td>
</tr>
<tr>
<td>PIN_DEBIT_TRANS_DISALLWD</td>
<td>Non-USD pin debit transactions are not supported. PINNED Debit for USD transactions only.</td>
</tr>
<tr>
<td>PLAN_IN_USE</td>
<td>This plan has already been used to generate commission calculations and can't be deleted.</td>
</tr>
<tr>
<td>PLAN_OVERLAP_DISALLWD</td>
<td>Plan overlap is not permitted. You have attempted to assign someone to this plan for a time period that overlaps with another plan.</td>
</tr>
<tr>
<td>PMT_ALREADY_APPRVD</td>
<td>The payment has already been approved and sent to the bill pay carrier for processing.</td>
</tr>
<tr>
<td>PMT_ALREADY_EXISTS</td>
<td>A payment with the same amount and date already exists for this payee.</td>
</tr>
<tr>
<td>PMT_ALREADY_SBMTD</td>
<td>This payment has already been submitted for online bill pay.</td>
</tr>
<tr>
<td>PMT_EDIT_DISALLWD</td>
<td>Access to this Bill Pay transaction is restricted, and it cannot be modified. Transactions can only be modified until 3PM CST on the payment date.</td>
</tr>
<tr>
<td>PMT_EDIT_DISALLWD</td>
<td>This liability payment cannot be edited while it has an Automated Clearing House transmission in process. &lt;/TD&gt;&lt;/TR&gt;&lt;TR class=text&gt;&lt;/TD&gt;&lt;/TR&gt;&lt;TR class=text&gt; To view the status of payments with ACH transmissions, go to Transactions &gt; View ACH Vendor Payments Status.</td>
</tr>
<tr>
<td>POSITIVE_BIN_QTY_REQD</td>
<td>Bin quantities must be positive or zero.</td>
</tr>
<tr>
<td>POSITIVE_QTY_REQD</td>
<td>Assembly member items must have positive quantities.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>POSITIVE_UNITCOST_REQD</td>
<td>Lines cannot have a negative unit cost.</td>
</tr>
<tr>
<td>POSTING_DISCOUNT_DISALLWD</td>
<td>Posting Discounts are not allowed on lines with Revenue Recognition Schedules.</td>
</tr>
<tr>
<td>POSTING_PRD_SETUP_REQD</td>
<td>Creation of Journal Entries require a single Accounting Period value across all Revenue Recognition events. Please setup a 'Posting Period' filter.</td>
</tr>
<tr>
<td>PRDS_DISALLWD_NAMES_NOT_UNIQUE</td>
<td>After adding new periods, not all names would be unique.</td>
</tr>
<tr>
<td>PRD_SETUP_REQD</td>
<td>You must change your period definitions to contain fiscal years. Please visit 'Setup-&gt;Manage Accounting Periods' and click 'Set Up Year'.</td>
</tr>
<tr>
<td>PRD_SETUP_REQD</td>
<td>You must define the periods of the prior fiscal year. Please visit 'Setup-&gt;Manage Accounting Periods' and click 'Set Up Year'.</td>
</tr>
<tr>
<td>PRDS_DISALLWD_NAMES_NOT_UNIQUE</td>
<td>After adding new periods, not all names would be unique.</td>
</tr>
<tr>
<td>PREF_VENDOR_COST_REQD</td>
<td>Drop ship/Special Order items must have a preferred vendor and a purchase price.</td>
</tr>
<tr>
<td>PREF_VENDOR_REQD</td>
<td>Drop ship/Special Order items must have a preferred vendor for each of the (1) the item is accessible to.</td>
</tr>
<tr>
<td>PREFERRED_TAX_AGENCY_REQD</td>
<td>A preferred Tax Agency has been deleted - Please choose a new one in &lt;a href=&quot;/app/setup/acctsetup.nl&quot;&gt;Set Up Accounting&lt;/a&gt;</td>
</tr>
<tr>
<td>PREFERRED_TAX_AGENCY_REQD</td>
<td>Error: No preferred Tax Agencies have been set up (go to &lt;a href=&quot;/app/setup/acctsetup.nl&quot;&gt;Set Up Accounting&lt;/a&gt;)</td>
</tr>
<tr>
<td>PRIVATE_RCRD_ACCESS_DISALLWD</td>
<td>You cannot view or edit this record because it is marked private</td>
</tr>
<tr>
<td>PRIVATE_STATUS_CHNG_DISALLWD</td>
<td>You cannot make this contact private.</td>
</tr>
<tr>
<td>PSWD_EXPIRED</td>
<td>Password has expired. Please change your NetSuite password before continuing.</td>
</tr>
<tr>
<td>PSWD_REQD</td>
<td>A password must be entered when granting login access privileges to this record.</td>
</tr>
<tr>
<td>PSWD_REQD</td>
<td>Password is empty.</td>
</tr>
<tr>
<td>PSWD_REQD</td>
<td>Please type your password into both fields.</td>
</tr>
<tr>
<td>PSWD_REQD</td>
<td>You must provide a password to give this person access to your account.</td>
</tr>
<tr>
<td>PWSDS_DONT_MATCH</td>
<td>New passwords don't match.</td>
</tr>
<tr>
<td>PWSDS_DONT_MATCH</td>
<td>The passwords you entered do not match. Please reenter your passwords.</td>
</tr>
<tr>
<td>PWSDS_DONTMATCH</td>
<td>The passwords you entered do not match. Please reenter your passwords.</td>
</tr>
<tr>
<td>PWSDS_DONT_MATCH</td>
<td>The passwords you have entered do not match.</td>
</tr>
<tr>
<td>QTY_EXCEEDED_QTY_BUCKETS</td>
<td>More quantities defined than there are quantity buckets</td>
</tr>
<tr>
<td>QTY_REQD</td>
<td>Quantities must be defined</td>
</tr>
<tr>
<td>RATE_REQUEST_SHPMNT_REQD</td>
<td>The rate request shipment value has not been set.</td>
</tr>
<tr>
<td>RATE_SRVC_UNAVAILBL</td>
<td>The rate for this service is not available for the specified source and destination addresses.</td>
</tr>
<tr>
<td>RCRD_DELETED_SINCE_RETRIEVED</td>
<td>Items you have requested in the record have been deleted since you retrieved the form</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RCRD_DELETED_SINCE_RETRIEVED</td>
<td>The record has been deleted since you retrieved it.</td>
</tr>
<tr>
<td>RCRD_DSNT_EXIST</td>
<td>Group Record Not Found</td>
</tr>
<tr>
<td>RCRD_DSNT_EXIST</td>
<td>That record does not exist. (1)</td>
</tr>
<tr>
<td>RCRD_DSNT_EXIST</td>
<td>There are no records of this type.</td>
</tr>
<tr>
<td>RCRD_EDITED_SINCE_RETRIEVED</td>
<td>The record has been edited since you retrieved it. Hit the back button and click Refresh/Reload to retrieve the updated record, then resubmit your changes.</td>
</tr>
<tr>
<td>RCRD_HAS_BEEN_CHANGED</td>
<td>Record has been changed</td>
</tr>
<tr>
<td>RCRD_ID_NOT_INT</td>
<td>Record id is not integer: (1)</td>
</tr>
<tr>
<td>RCRD_LOCKED_BY_WF</td>
<td>This record has been locked by a user defined workflow.</td>
</tr>
<tr>
<td>RCRD_NOT_FOUND</td>
<td>Could not find any records by this name.</td>
</tr>
<tr>
<td>RCRD_PREVSLY_DELETED</td>
<td>This record has already been deleted.</td>
</tr>
<tr>
<td>RCRD_PREVSLY_DELETED</td>
<td>This record has been deleted since the list was generated.</td>
</tr>
<tr>
<td>RCRD_REF_RCRD_TYP_MISMATCH</td>
<td>The record type and its object reference are not matched.</td>
</tr>
<tr>
<td>RCRD_SUB_MISMATCH_WITH_CLASS</td>
<td>The subsidiary restrictions on this record are incompatible with those defined for class: (1). Subsidiary access on this record must be a subset of those permitted by the class.</td>
</tr>
<tr>
<td>RCRD_TYPE_REQD</td>
<td>The record type is required.</td>
</tr>
<tr>
<td>RCRD_UNEDITABLE</td>
<td>That record is not editable.</td>
</tr>
<tr>
<td>REACHED_LIST_END</td>
<td>You have reached the end of the list.</td>
</tr>
<tr>
<td>REACHED_LIST_START</td>
<td>You have reached the start of the list.</td>
</tr>
<tr>
<td>REC_TYP_REQD</td>
<td>You must provide either standard or custom record type information.</td>
</tr>
<tr>
<td>RECALCING_PLAN_SCHEDUL</td>
<td>Cannot inactivate a plan when schedules in the plan are recalculating. Try again when recalculation is complete.</td>
</tr>
<tr>
<td>RECUR_EVENT_DISALLWD</td>
<td>A yearly event cannot be on the 29th of February</td>
</tr>
<tr>
<td>RECURS_TV_DISPALLWD</td>
<td>ERROR: Recursive Reference</td>
</tr>
<tr>
<td>REPORT_EXPIRED</td>
<td>Your report request has expired. A newly created report definition will normally expire after 15 minutes of inactivity unless the definition is saved. Click back to the report definition page and re-submit your report request.</td>
</tr>
<tr>
<td>REQD_FORM_TAG_MISSING</td>
<td>Your HTML template file is missing some mandatory fields or a form tag. Please make the changes to the form and try again</td>
</tr>
<tr>
<td>REQD_FORM_TAG_MISSING</td>
<td>Your Online HTML Form template is missing a required closing FORM tag</td>
</tr>
<tr>
<td>REQD_FORM_TAG_MISSING</td>
<td>Your Online HTML Form template is missing one or more of the required following required tags: &lt;HTML&gt;, &lt;HEAD&gt;</td>
</tr>
<tr>
<td>REQD_FORM_TAG_MISSING</td>
<td>Your Online HTML template is missing the required form tag for the main form: &lt;NLFORM&gt;</td>
</tr>
<tr>
<td>REQD_LOC_FIELDS_MISSING</td>
<td>Location (1) does not have all required fields set. Please ensure the State, Zip/Postal Code, and Country fields are set and try your request again.</td>
</tr>
<tr>
<td>REQD_SUB_FIELDS_MISSING</td>
<td>The Subsidiary (1) does not have all required fields set. Please ensure the State, Zip/Postal Code, and Country fields are set and try your request again.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>REQUEST_PARAM_REQD</td>
<td>This request is missing a required parameter.</td>
</tr>
<tr>
<td>REV_REC_DATE_REQD</td>
<td>No Revenue Recognition Start Date Specified</td>
</tr>
<tr>
<td>REV_REC_TMPLT_DATA_MISSING</td>
<td>One or more line items on this transaction have Variable Revenue Recognition Templates, but do not have the required (1) also populated. Please either change the Template for these items or indicate which (1) will be used to schedule the recognition of revenue.</td>
</tr>
<tr>
<td>REV_REC_UPDATE_DISALLOWED</td>
<td>Modification of revenue recognition related information on this item is not allowed because revenue has been recognized for this or related lines.</td>
</tr>
<tr>
<td>REVERSAL_DATE_WARNING</td>
<td>Reversal Date is in a closed accounting period. Please go to Manage Accounting Periods and re-open the accounting period.</td>
</tr>
<tr>
<td>ROLE_REQD</td>
<td>Please specify a role to which access should be granted</td>
</tr>
<tr>
<td>ROLE_REQUIRED</td>
<td>To login, a role is required unless a default has been previously set.</td>
</tr>
<tr>
<td>ROUNGING_DIFF_TOO_BIG</td>
<td>rounding difference too big (-) tax1: (1) /tax2: (2)</td>
</tr>
<tr>
<td>ROUNGING_ERROR</td>
<td>Rounding Error: (1)</td>
</tr>
<tr>
<td>ROUTING_NUM_REQD</td>
<td>Missing Routing Number. Bank routing number is a required field and it cannot be null or empty.</td>
</tr>
<tr>
<td>SALES_DISCOUNT_ACCT_REQD</td>
<td>Please set the Sales Discount Account Preference</td>
</tr>
<tr>
<td>SAME_ACCT_TYP_REQD_FOR_PARENT</td>
<td>Parent account must be of same account type.</td>
</tr>
<tr>
<td>SAVED_SRCH_EMAIL_ERROR</td>
<td>E-mail Alert Failure using saved search '{1}'. Reason: (2)</td>
</tr>
<tr>
<td>SCHEDUL_EDIT_DISALLOWED</td>
<td>This schedule cannot be edited as it has already been used for commission calculations. Please go back and select 'save as new' instead.</td>
</tr>
<tr>
<td>SCHEDULED_REPORT_ERROR</td>
<td>NetSuite encountered an error while trying to generate your scheduled report.</td>
</tr>
<tr>
<td>SCHEDULED_REPORT_ERROR</td>
<td>NetSuite encountered an error while trying to generate your scheduled report. Please click {1}here{2} to notify NetSuite Support.</td>
</tr>
<tr>
<td>SCHEDULED_REPORT_ERROR</td>
<td>NetSuite encountered an error while trying to generate your scheduled report. This error most likely occurred because the query was too large. To run this report successfully, try using filters, or select a different date/period range to narrow the query.</td>
</tr>
<tr>
<td>SCHEDULED_REPORT_ERROR</td>
<td>NetSuite encountered an error while trying to generate your scheduled report. This error most likely occurred because the query was too large. To run this report successfully, try using filters, or select a different date/period range to narrow the query. If you continue to encounter errors, please click {1}here{2} to notify NetSuite Support.</td>
</tr>
<tr>
<td>SEARCH_DATE_FILTER_REQD</td>
<td>The search must have a date column as an available filter</td>
</tr>
<tr>
<td>SEARCH_ERROR</td>
<td>Unable to search - Unexpected error in search engine.</td>
</tr>
<tr>
<td>SEARCH_INTEGER_REQD</td>
<td>Please enter an integer number to search on.</td>
</tr>
<tr>
<td>SEARCH_TIMED_OUT</td>
<td>Your search has timed out. If your search includes the '{1}' operator, try using '{2}' instead. If your search includes broad search criteria, try narrowing the criteria.</td>
</tr>
<tr>
<td>SEARCH_TIMED_OUT</td>
<td>Your search has timed out. This might be avoided by using a smaller page size.</td>
</tr>
<tr>
<td>SECURE_TRANS_REQD_ON_CHECKOUT</td>
<td>Store Server Error: As configured, this server does not permit secure transactions, required by store checkout.</td>
</tr>
<tr>
<td>SERIAL_NUM_MATCH_MULTI_ITEMS</td>
<td>(1) different items match this serial number. Select an item from the item dropdown list.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SESSION_TERMD_2NDLOGIN_DECTD</td>
<td>You can have a maximum of {1} active users at a time that are enabled for Offline Sales Client. If you would like to add active users, please contact your account manager to discuss your upgrade options. Or, you may choose to disable Offline Sales Client on an existing user before enabling a new one.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>SSS_DRIP_EMAIL_RAN_OUT_OF_COUPON_CODES</td>
<td>Campaign event ran out of coupon codes.</td>
</tr>
<tr>
<td>SSS_DUP_DRIP_EMAIL</td>
<td>You are attempting to send the same campaign email twice to the same recipient.</td>
</tr>
<tr>
<td>SSS_FILE_SIZE_EXCEEDED</td>
<td>The file you are trying to load exceeds the maximum allowed file size of {1} megabyte.</td>
</tr>
<tr>
<td>SSS_INSTRUCTION_COUNT_EXCEEDED</td>
<td>Script Execution Instruction Count Exceeded.</td>
</tr>
<tr>
<td>SSS_INVALID_ATTACHMENT_RECORD_TYPE</td>
<td>Attaching of record type {1} to {2} is not supported.</td>
</tr>
<tr>
<td>SSS_INVALID_BCC_EMAIL</td>
<td>One or more bcc emails are not valid.</td>
</tr>
<tr>
<td>SSS_INVALID_CC_EMAIL</td>
<td>One or more cc emails are not valid.</td>
</tr>
<tr>
<td>SSS_INVALID_CMPGN_EVENT_ID</td>
<td>That campaign event is invalid, disabled, or is not a Lead Nurturing Email event. Please select an active Lead Nurturing Email campaign event.</td>
</tr>
<tr>
<td>SSS_INVALID_EMAIL_TEMPLATE</td>
<td>That email template is invalid, disabled, or no longer exists. Please select an active email template.</td>
</tr>
<tr>
<td>SSS_INVALID_FORM_ELEMENT_NAME</td>
<td>You have entered an invalid form element name. It must be prefixed with “custpage”, unique, and cannot contain any non-alphanumeric characters to be added to the form or sublist.</td>
</tr>
<tr>
<td>SSS_GSO_FLTR_OPRTOR</td>
<td>SSS Invalid GetSelectOption Filter Operator {1}, must be one of the following: {2}</td>
</tr>
<tr>
<td>SSS_INVALID_HEADER</td>
<td>One or more headers are not valid.</td>
</tr>
<tr>
<td>SSS_INVALID_HOST_CERT</td>
<td>An untrusted, unsupported, or invalid certificate was found for this host.</td>
</tr>
<tr>
<td>SSS_INVALID_LIST_COLUMN_NAME</td>
<td>You have entered an invalid list column name. It must be unique and cannot contain any non-alphanumeric characters.</td>
</tr>
<tr>
<td>SSS_INVALID_LOCK_WAIT_TIME</td>
<td>You have entered an invalid wait time ({1}) for acquiring a lock. The wait time must be greater than 0ms and less than 5000ms.</td>
</tr>
<tr>
<td>SSS_INVALID_LOG_TYPE</td>
<td>Execution log type must be one of AUDIT, DEBUG, ERROR, or EMERGENCY.</td>
</tr>
<tr>
<td>SSS_INVALID_PORTLET_INTERVAL</td>
<td>You have entered an invalid refresh interval of {1} seconds. It must not be negative.</td>
</tr>
<tr>
<td>SSS_INVALID_PORTLET_INTERVAL</td>
<td>You have entered an invalid refresh interval of {1} seconds. It must be at least {2} seconds if not in testing status.</td>
</tr>
<tr>
<td>SSS_INVALID_SCRIPTLET_ID</td>
<td>That Suitelet is invalid, disabled, or no longer exists.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_COL</td>
<td>An nlobjSearchColumn contains an invalid column, or is not in proper syntax: {1}.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_COLUMN_JOIN</td>
<td>An nlobjSearchColumn contains an invalid column join ID, or is not in proper syntax: {1}.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_COLUMN_SUM</td>
<td>An nlobjSearchColumn contains an invalid column summary type, or is not in proper syntax: {1}.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_FILTER</td>
<td>An nlobjSearchFilter contains invalid search criteria: {1}.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_FILTER_JOIN</td>
<td>An nlobjSearchFilter contains an invalid join ID, or is not in proper syntax: {1}.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_OPERATOR</td>
<td>An nlobjSearchFilter contains an invalid operator, or is not in proper syntax: {1}.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>You have attempted an invalid sublist or line item operation. You are either trying to access a field on a non-existent line or you are trying to add or remove lines from a static sublist.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBMIT_OPTION</td>
<td>You have entered an invalid submit option for this record type: (1)</td>
</tr>
<tr>
<td>SSS_INVALID_TYPE_ARG</td>
<td>You have entered an invalid type argument: (1)</td>
</tr>
<tr>
<td>SSS_INVALID_UI_OBJECT_TYPE</td>
<td>That operation is not supported for this type of UI object: (1). It is only supported for type: (2).</td>
</tr>
<tr>
<td>SSS_INVALID_URL</td>
<td>The URL must be a fully qualified HTTP or HTTPS URL if it is referencing a non-NetSuite resource.</td>
</tr>
<tr>
<td>SSS_INVALID_URL_CATEGORY</td>
<td>The URL category must be one of RECORD, TASKLINK or SUITELET.</td>
</tr>
<tr>
<td>SSS_INVALID_WF_RCRD_TYPE</td>
<td>You have entered an invalid record type {1}. Workflow automation is not supported for this type of record.</td>
</tr>
<tr>
<td>SSS_MEMORY_USAGE_EXCEEDED</td>
<td>Script Out Of Memory.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>{1}: Missing a required argument: {2}</td>
</tr>
<tr>
<td>SSS_QUEUE_LIMIT_EXCEEDED</td>
<td>Script Queue Usage Limit Exceeded</td>
</tr>
<tr>
<td>SSS_RECORD_TYPE_MISMATCH</td>
<td>The record you are attempting to load has a different type: (1) from the type specified: (2).</td>
</tr>
<tr>
<td>SSS_REQUEST_LIMIT_EXCEEDED</td>
<td>Request Limit Exceeded!</td>
</tr>
<tr>
<td>SSS_REQUEST_TIME_EXCEEDED</td>
<td>The host you are trying to connect to is not responding or has exceeded the maximum allowed response time.</td>
</tr>
<tr>
<td>SSS_SCRIPT_DESERIALIZATION_FAILURE</td>
<td>Scheduled Script deserialization failure</td>
</tr>
<tr>
<td>SSS_SEARCH_TIMEOUT</td>
<td>Your search request has timed out. You may need to refine your search or combine the results of multiple searches to achieve the desired result.</td>
</tr>
<tr>
<td>SSS_SSO_CONFIG_REQD</td>
<td>The SuiteSignOn object (1) is not configured for use with this script. You must specify the script as a connection point for this SuiteSignOn.</td>
</tr>
<tr>
<td>SSS_STACK_FRAME_DEPTH_EXCEEDED</td>
<td>Script Stack Frame Depth Exceeded</td>
</tr>
<tr>
<td>SSS_TIME_LIMIT_EXCEEDED</td>
<td>Script Execution Time Exceeded.</td>
</tr>
<tr>
<td>SSS_TRANS_IN_PROGRESS</td>
<td>You cannot call nlapiBeginTransaction() because there is already a transaction in progress.</td>
</tr>
<tr>
<td>SSS_TRANSACTION_REQD</td>
<td>That operation can only be performed when there is a transaction in progress.</td>
</tr>
<tr>
<td>SSS_UNKNOWN_HOST</td>
<td>The host you requested (1) is unknown or cannot be found.</td>
</tr>
<tr>
<td>SSS_USAGE_LIMIT_EXCEEDED</td>
<td>Script Execution Usage Limit Exceeded</td>
</tr>
<tr>
<td>START_DATE_AFTERT_END_DATE</td>
<td>The start date must precede the end date.</td>
</tr>
<tr>
<td>START_DATE_REQD</td>
<td>Please enter a value for (1) Start Date</td>
</tr>
<tr>
<td>STATE_ALREADY_EXISTS</td>
<td>A State/Province/County with the same name already exists.</td>
</tr>
<tr>
<td>STATE_REQD</td>
<td>State is a required field and it cannot be null or empty.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>STATUS_ASSIGNEE_REQD</td>
<td>The issue status (1) does not define an assignee issue role. That status may not be used until this is corrected.</td>
</tr>
<tr>
<td>STORAGE_LIMIT_EXCEEDED</td>
<td>You entered a value that will exceed the internal storage limit of (1). Please reduce the number.</td>
</tr>
<tr>
<td>STORE_ALIAS_UNAVAILABLE</td>
<td>The Store alias you chose &quot;(1)&quot; is already taken. Please go back and choose another.</td>
</tr>
<tr>
<td>STORE_DOMAIN_UNAVAILABLE</td>
<td>The store domain name you chose '{1}' is already taken. Please go back and choose another.</td>
</tr>
<tr>
<td>SUB_MISMATCH</td>
<td>Resources for Time and Material projects must be in the customer's subsidiary. The following resources are in a different subsidiary: {1}</td>
</tr>
<tr>
<td>SUB_RESTRICT_VIEW_REQD</td>
<td>Subsidiary Restrict View Required: You must first restrict your view to a subsidiary before running this report (Home &gt; Set Preferences &gt; Restrict View&lt;/a&gt;).</td>
</tr>
<tr>
<td>SUB_TAX_AGENCY_REQD</td>
<td>No tax agency defined for subsidiary</td>
</tr>
<tr>
<td>SUBITEM_REQD</td>
<td>You must first select the new subitems on the matrix tab you want to add.</td>
</tr>
<tr>
<td>SUBITEM_REQD</td>
<td>You must first select the subitems on the matrix tab you want to create.</td>
</tr>
<tr>
<td>SUBSIDIARY_MISMATCH</td>
<td>The employee and billable customer must be in the same subsidiary.</td>
</tr>
<tr>
<td>SUCCESS_TRANS</td>
<td>The transaction was entered (1) successfully, (2)</td>
</tr>
<tr>
<td>SUPRT_CNTR_LOGIN_ERROR</td>
<td>{1} Support Center login error: we are unable to find the customer record for account={2}</td>
</tr>
<tr>
<td>TAG_ALREADY_EXISTS</td>
<td>That tag name is already being used. Please go back and rename it.</td>
</tr>
<tr>
<td>TAX_ACCT_SETUP_REQD</td>
<td>Tax Accounts Not Defined.</td>
</tr>
<tr>
<td>TAX_CODE_REQD</td>
<td>No default tax code is defined for country {1}</td>
</tr>
<tr>
<td>TAX_CODES_SETUP_PROBLEM</td>
<td>The tax codes haven't been set properly</td>
</tr>
<tr>
<td>TAX_CODES_SETUP_REQD</td>
<td>Can't open store for (1). This company does not have its tax codes fully set up. This is required to properly calculate taxes on international, other-province and same-province orders.</td>
</tr>
<tr>
<td>TAX_GROUP_SETUP_REQD</td>
<td>You have not created tax groups in your NetSuite account. To ensure that your customers are charged the correct amount of sales tax, you must create tax groups by entering them manually at Lists &gt; Accounting &gt; Tax Groups &gt; New.</td>
</tr>
<tr>
<td>TAX_PRD_REQD</td>
<td>No Current Tax Period is defined. &lt;a href='/app/setup/period/taxperiods.nl'&gt;Click here&lt;/a&gt; to create a tax period.</td>
</tr>
<tr>
<td>TAX_SETUP_REQD</td>
<td>The tax period range {1} has not been defined. Please visit '&lt;A href='{2}'&gt;Setup &gt; Accounting &gt; Manage Tax Periods&lt;/A&gt;' to define this period or set up your year.</td>
</tr>
<tr>
<td>TEMPLATE_NOT_FOUND</td>
<td>Template not found</td>
</tr>
<tr>
<td>THIRD_PARTY_BILLING_ACCT_REQD</td>
<td>A 3rd Party Billing Account Number must be provided when selecting a 3rd Party Billing Type.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TICKET_NOT_LOCATED</td>
<td>The ticket {1} cannot be located in the error database. If this is from a customer logged case, the error may not yet be inserted into the system.</td>
</tr>
<tr>
<td>TIME_ENTRY_DISALLWD</td>
<td>{1} does not allow time entry.</td>
</tr>
<tr>
<td>TIMEOUT_THE_RECORD_DOESNT_EXIST_ANYMORE</td>
<td>Timeout: the record doesn’t exist anymore.</td>
</tr>
<tr>
<td>TOPIC_REQD</td>
<td>You must select and add a topic to this solution.</td>
</tr>
<tr>
<td>TRAN_DATE_REQD</td>
<td>Missing transaction date.</td>
</tr>
<tr>
<td>TRAN_LINE_FX_AMT_REQD</td>
<td>Missing foreign currency amount on non-variance transaction line</td>
</tr>
<tr>
<td>TRAN_LINK_FX_AMT_REQD</td>
<td>Missing foreign currency amount on non-variance transaction link</td>
</tr>
<tr>
<td>TRAN_PERIOD_CLOSED</td>
<td>The action is causing generation of foreign exchange variance in a closed period. Please retry with {1} as the exchange rate.</td>
</tr>
<tr>
<td>TRAN_PERIOD_CLOSED</td>
<td>You are not allowed to change the revenue recognition status for one or more lines on this transaction as it would impact a closed period.</td>
</tr>
<tr>
<td>TRAN_PRD_CLOSED</td>
<td>You cannot change the G/L impact of a transaction in a closed period.</td>
</tr>
<tr>
<td>TRANS_ALREADY_REFUNDED</td>
<td>Transaction Already refunded. A refund has already been performed on the transaction.</td>
</tr>
<tr>
<td>TRANS_ALREADY_SETTLED</td>
<td>Transaction Already Settled. Void failed because transaction has already settled, submit credit.</td>
</tr>
<tr>
<td>TRANS_ALREADY_VOIDED</td>
<td>Transaction Already Voided. Void failed because transaction is already voided.</td>
</tr>
<tr>
<td>TRANS_AMTS_UNBALNCD</td>
<td>Transaction is not in balance! amounts+taxes+shipping: (1), total amount: (2)</td>
</tr>
<tr>
<td>TRANS_APPLIED_AMTS_UNBALNCD</td>
<td>Transaction is not in balance! Total to apply of $1 does not equal sum of applied $2 and unapplied $3</td>
</tr>
<tr>
<td>TRANS_APPLIED_AMTS_UNBALNCD</td>
<td>Transaction is not in balance! Total to apply of $1 does not equal sum of payment $2 and credits $3 and deposits $4</td>
</tr>
<tr>
<td>TRANS_CLASS_UNBALNCD</td>
<td>Transaction out of balance for class (1) total = (2).</td>
</tr>
<tr>
<td>TRANS_DEPT_UNBALNCD</td>
<td>Transaction out of balance for department (1) total = (2).</td>
</tr>
<tr>
<td>TRANS_DOES_NOT_EXIST</td>
<td>No transaction exists for that entity.</td>
</tr>
<tr>
<td>TRANS_DSNT_EXIST</td>
<td>The transaction you are attempting to access does not exist.</td>
</tr>
<tr>
<td>TRANS_EDIT_DISALLWD</td>
<td>This transaction is in a period that has been closed. You may not edit it.</td>
</tr>
<tr>
<td>TRANS_EDIT_DISALLWD</td>
<td>You cannot edit this transaction. {1} does not support the imported transaction.</td>
</tr>
<tr>
<td>TRANS_FORGN_CRNCY_MISMATCH</td>
<td>Transaction and foreign currency account use different currencies.</td>
</tr>
<tr>
<td>TRANS_FORGN_CUR_UNBALNCD</td>
<td>Transaction was not in balance (Foreign currency). Posting total = {1}</td>
</tr>
<tr>
<td>TRANS_FORGN_CUR_UNBALNCD</td>
<td>Transaction was not in balance (Foreign currency). Total = {1}</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TRANS_IN_USE</td>
<td>This transaction cannot be deleted because it is linked to by one or more transactions. Click <code>&lt;a href='/app/accounting/transactions/payments.nl?id={1}&amp;label={2}&amp;type={3}&amp;alllinks=T'&gt;here&lt;/a&gt;</code> to see the list of linked transactions.</td>
</tr>
<tr>
<td>TRANS_LINE_AND_PMT_UNBALNCD</td>
<td>Transaction is not in balance! Line item sum of $(1) not equal to payment amount $(2)</td>
</tr>
<tr>
<td>TRANS_LINES_UNBALNCD</td>
<td>Transaction is not in balance! Line item sum of $(1) does not equal amount of $(2)</td>
</tr>
<tr>
<td>TRANS_LINES_UNBALNCD</td>
<td>Transaction is not in balance! Line item sum of $(1) does not equal applied amount of $(2)</td>
</tr>
<tr>
<td>TRANS_LOC_UNBALNCD</td>
<td>Transaction out of balance for location (1) total = (2).</td>
</tr>
<tr>
<td>TRANS_NOT_CLEANED</td>
<td>Transaction not cleaned up.</td>
</tr>
<tr>
<td>TRANS_NOT_COMPLETED</td>
<td>Transaction was not complete.</td>
</tr>
<tr>
<td>TRANS_PROCESSING_ERROR</td>
<td>Errors occurred while processing the selected transaction. Please process it individually for more information.</td>
</tr>
<tr>
<td>TRANS_UNBALNCD</td>
<td>The debits and credits are not balanced on this transaction because amounts entered include more decimal places than are supported for this currency. Please round off the amount of each line to a maximum of (1) decimal places.</td>
</tr>
<tr>
<td>TRANS_UNBALNCD</td>
<td>Transaction is not in balance! $(1)$</td>
</tr>
<tr>
<td>TRANS_UNBALNCD</td>
<td>Transaction is not in balance! $(1),(2)$ othercount = $(3)$</td>
</tr>
<tr>
<td>TRANS_UNBALNCD</td>
<td>Transaction out of balance for $(1) $(2)$ total = $(3)$.</td>
</tr>
<tr>
<td>TRANS_UNBALNCD</td>
<td>Transaction was not in balance. Posting total = $(1)$</td>
</tr>
<tr>
<td>TRANS_UNBALNCD</td>
<td>Transaction was not in balance. Total = $(1)$</td>
</tr>
<tr>
<td>TRANSACTION_DELETED</td>
<td>The transaction you are attempting to access has been deleted.</td>
</tr>
<tr>
<td>TRANSORD_SHIP_REC_MISMATCH</td>
<td>You can not receive more from a transfer order than you have shipped.</td>
</tr>
<tr>
<td>TWO_FA_AUTH_REQD</td>
<td>All your other roles require a one-time key at login. Please click &quot;Go Back&quot; to enter a one-time key or contact the company’s administrator if you have questions.</td>
</tr>
<tr>
<td>TWO_FA_REQD</td>
<td>Two-Factor Authentication required</td>
</tr>
<tr>
<td>UNABLE_TO_PRINT_CHECKS</td>
<td>Unable to print checks.</td>
</tr>
<tr>
<td>UNABLE_TO_PRINT_DEPOSITS</td>
<td>Unable to print deposits.</td>
</tr>
<tr>
<td>UNAUTH_CAMPAIGN_RSPNS_RQST</td>
<td>Unauthorized campaign response request</td>
</tr>
<tr>
<td>UNAUTH_UNSUBSCRIBE_RQST</td>
<td>Unauthorized unsubscribe request</td>
</tr>
<tr>
<td>UNDEFINED_ACCTNG_PRD</td>
<td>The accounting period range (1) has not been defined. Please visit <code>&lt;A href='app/setup/period/fiscalperiods.nl'&gt;Setup &gt; Accounting &gt; Manage Accounting Periods&lt;/A&gt;</code> to define this period or set up your year.</td>
</tr>
<tr>
<td>UNDEFINED_ACCTNG_PRD</td>
<td>The comparison accounting period range (1) has not been defined. Please visit <code>&lt;A href='app/setup/period/fiscalperiods.nl'&gt;Setup &gt; Accounting &gt; Manage Accounting Periods&lt;/A&gt;</code> to define this period or set up your year.</td>
</tr>
<tr>
<td>UNDEFINED_ACCTNG_PRD</td>
<td>The default accounting period for this report has not been defined. Please visit <code>&lt;A href='app/setup/period/fiscalperiods.nl'&gt;Setup &gt; Accounting &gt; Manage Accounting Periods&lt;/A&gt;</code> to define this period or set up your year.</td>
</tr>
<tr>
<td>UNDEFINED_CSTM_FIELD</td>
<td>Undefined customfield.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>UNDEFINED_TAX_PRD</td>
<td>The default tax period for this report has not been defined. Please visit '&lt;A href='/app/setup/period/taxperiods.nl'&gt;Setup &gt; Accounting &gt; Manage Tax Periods&lt;/A&gt;' to define this period or set up your year.</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>An error occurred while processing item options.</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>An unexpected error has occurred.</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>An unexpected error has occurred. A FedEx Shipping Label was not generated.</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>An unexpected error has occurred while generating this content. Our Customer Support staff have been notified and are looking into the problem.</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>An unexpected error has occurred while synching a record. Click [OK] to skip the record and continue.</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>An unexpected error has occurred. NetSuite Customer Support has been alerted to this problem.</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>An unexpected error occurred while extracting email from SMTP server</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>An unexpected error occurred while logging email request completion</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>An unexpected error occurred while logging email request failure</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>An unexpected error occurred while logging email request start</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>An unexpected error occurred while processing the payment.</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>An unexpected error occurred with the group SQL</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>An Unexpected JavaScript Error has occurred</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>Error</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>Error: (1)</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>Please specify an scompid</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>Problem during commission calculation</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>An unexpected error occurred.</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>Dto java class is not defined for (1).</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>Server error: no dto class is defined for record of type (1)</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>Server error: missing database entries in WSRecordElement and WSNameSpace table for object of (1)</td>
</tr>
<tr>
<td>UNEXPECTED_ERROR</td>
<td>Application error: no form request class is defined for record of type (1)</td>
</tr>
<tr>
<td>UNIQUE_CONTACT_NAME_REQD</td>
<td>Contact names must be unique</td>
</tr>
<tr>
<td>UNIQUE_CUST_EMAIL_REQD</td>
<td>A customer record with this email address already exists. You must enter a unique customer email address for each record you create.</td>
</tr>
<tr>
<td>UNIQUE_CUST_EMAIL_REQD</td>
<td>A customer record with this email address already exists. You must enter a unique customer email address for each record you create. To correct this record, click &lt;a href=&quot;javascript:history.go(-1);&quot;&gt;back&lt;/a&gt; and enter a new customer email address in the Customer field. Then, click Submit.</td>
</tr>
<tr>
<td>UNIQUE_CUST_ID_REQD</td>
<td>A customer record with this ID already exists. You must enter a unique customer ID for each record you create.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>UNIQUE_CUST_ID_REQD</td>
<td>A customer record with this ID already exists. You must enter a unique customer ID for each record you create. To correct this record, click <code>&lt;a href='javascript:history.go(-1);'&gt;back&lt;/a&gt;</code> and enter a new customer ID in the Customer field. Then, click Submit.</td>
</tr>
<tr>
<td>UNIQUE_ENTITY_NAME_REQD</td>
<td>multiple sub-customers or jobs have name <code>{1}</code> which would create a naming conflict upon merge. All names must be unique. Before merging, you must change one of the sub named <code>{2}</code> to something else.</td>
</tr>
<tr>
<td>UNIQUE_GROUPID_REQD</td>
<td>You must specify exactly one numeric groupid</td>
</tr>
<tr>
<td>UNIQUE_PARTNER_CODE_REQD</td>
<td><code>{1: name of partner record}</code> Code &quot;{2: partner code}&quot; already exists. Please select a unique code for each record.</td>
</tr>
<tr>
<td>UNIQUE_QTY_REQD</td>
<td>Quantities must be unique</td>
</tr>
<tr>
<td>UNIQUE_RCRD_ID_REQD</td>
<td>A record with this ID already exists. You must enter a unique ID to create or update this record.</td>
</tr>
<tr>
<td>UNIQUE_SOLUTION_CODE_REQD</td>
<td>A solution with this particular solution code already exists. Please assign a different code.</td>
</tr>
<tr>
<td>UNITS_TYP_IN_USE</td>
<td>This units type is used by <code>{1}</code> <code>{2}</code>. You must delete the <code>{2}</code> and all associated transactions to delete this units type.</td>
</tr>
<tr>
<td>UNKNOWN_CARRIER</td>
<td>Package Tracking is not available for id <code>{1}</code>. Unknown carrier.</td>
</tr>
<tr>
<td>UNKNOWN_RCRD_TYPE</td>
<td>Unknown record type</td>
</tr>
<tr>
<td>UNKNOWN_SCRIPT_TYP</td>
<td>Unknown Script Type</td>
</tr>
<tr>
<td>UNKNWN_ALLOCCTN_SCHEDULER_FREQ_TYP</td>
<td>Unable to determine allocation schedule frequency type.</td>
</tr>
<tr>
<td>UNKNWN_EMAIL_AUTHOR</td>
<td>The author of this email cannot be found.</td>
</tr>
<tr>
<td>UNKNWN_EXCHANGE_RATE</td>
<td>Unable to determine the exchange rate for currency symbol <code>{1}</code>. Please create a currency item with an exchange rate for this currency.</td>
</tr>
<tr>
<td>UNRECOGNIZED_METHOD</td>
<td>unrecognized method '{1}'</td>
</tr>
<tr>
<td>UNSUBSCRIBE_REQD</td>
<td>Unsubscribe is mandatory, please enter a value for this field.</td>
</tr>
<tr>
<td>UNSUPPORTED_METHOD</td>
<td>Unsupported method</td>
</tr>
<tr>
<td>UNSUPPORTED_WS_VERSION</td>
<td>TBA not supported in this SuiteTalk endpoint version</td>
</tr>
<tr>
<td>UNSUPRTD_DOC_TYP</td>
<td>You attempted to upload an unsupported document type. Please try again with a selection from the list below:</td>
</tr>
<tr>
<td>UPDATE_DISALLWD</td>
<td>Update is not allowed</td>
</tr>
<tr>
<td>UPDATE_PRICE_AMT_REQD</td>
<td>Please specify an amount to update prices.</td>
</tr>
<tr>
<td>UPGRADE_WS_VERSION</td>
<td>Could not set '{1}' to field '{2}' of record number <code>{3}</code> due to schema enumeration restriction.</td>
</tr>
<tr>
<td>UPGRADE_WS_VERSION</td>
<td>Please consider upgrading to endpoint <code>{1}</code></td>
</tr>
<tr>
<td>UPGRADE_WS_VERSION</td>
<td>Sales order <code>&lt;id </code>{1}<code>&gt;</code> contains item serial/lot numbers that are not supported in your client application. You are not allowed to update serial/lot numbers on this sales order. Contact your software vendor for the latest SOAP web services upgrade.</td>
</tr>
<tr>
<td>UPGRADE_WS_VERSION</td>
<td>Sales order <code>&lt;id </code>{1}<code>&gt;</code> has items with more than one serial/lot numbers that is not supported in your client application. The serial/lot numbers have been removed to successfully return the sales order. Contact your software vendor for the latest SOAP web services upgrade.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>UPGRADE_WS_VERSION</td>
<td>This {1} has multiple {2}s. SOAP web services schema version (3) or greater is required to modify this {1}</td>
</tr>
<tr>
<td>UPGRADE_WS_VERSION</td>
<td>{1} (2) has multiple shipping routes enabled, which is only supported in version 2008_2 and newer. The shipping information has been omitted to successfully return this record.</td>
</tr>
<tr>
<td>UPS_CANT_INTEGRATE_FULFILL</td>
<td>The fulfillment cannot be integrated with UPS because the Shipping Integration Carrier is not set to UPS.</td>
</tr>
<tr>
<td>UPS_CONFIG_ERROR</td>
<td>A UPS configuration error occurred. Please contact tech support.</td>
</tr>
<tr>
<td>UPS_LICENSE_AGREEMENT_REQD</td>
<td>You must agree to the UPS license agreement</td>
</tr>
<tr>
<td>UPS_ONLINE_RATE_UNAVAILBL</td>
<td>The UPS Online Realtime Rates System is temporarily unavailable. Please resubmit your rate request in a few minutes.</td>
</tr>
<tr>
<td>UPS_ONLINE_RATE_UNAVAILBL</td>
<td>UPS did not return any rates for the specified origin and destination addresses.</td>
</tr>
<tr>
<td>UPS_ONLINE_SHIP_UNAVAILBL</td>
<td>The UPS Online Shipping System is temporarily unavailable. Please resubmit your fulfillment in a few minutes.</td>
</tr>
<tr>
<td>UPS_REG_NUM_IN_USE</td>
<td>The submitted UPS Registration Number, {1}, is already in use. Please resubmit the registration with a different UPS registration Number.</td>
</tr>
<tr>
<td>UPS_SETUP_REQD</td>
<td>No UPS registration was found. Please register your UPS account with NetSuite before attempting to send a fulfillment request to UPS.</td>
</tr>
<tr>
<td>UPS VOID_ERROR</td>
<td>The UPS Void failed due to a system failure.</td>
</tr>
<tr>
<td>UPS_XML_ERROR</td>
<td>XML Sent to UPS. UPS returned error code/text:</td>
</tr>
<tr>
<td>URL_ID_PARAM_REQD</td>
<td>URL is missing the id parameter. The file could not be retrieved.</td>
</tr>
<tr>
<td>URL_REQD</td>
<td>You must enter a URL for this media item.</td>
</tr>
<tr>
<td>USER CONTEXT REQD</td>
<td>User context is not set</td>
</tr>
<tr>
<td>USER_DISABLED</td>
<td>user disabled</td>
</tr>
<tr>
<td>USER_ERROR</td>
<td>Application ID mandatory but missing.</td>
</tr>
<tr>
<td>USER_ERROR</td>
<td>An error occurred during your last update.</td>
</tr>
<tr>
<td>USER_ERROR</td>
<td>A User Error Has Occurred</td>
</tr>
<tr>
<td>USER_ERROR</td>
<td>Detach requires an AttachBasicReference</td>
</tr>
<tr>
<td>USER_ERROR</td>
<td>Either internalId or externalId is required.</td>
</tr>
<tr>
<td>USER_ERROR</td>
<td>Folder cannot be made a subfolder of itself.</td>
</tr>
<tr>
<td>USER_ERROR</td>
<td>Gift Certificate From, Recipient Name, and Recipient Email are required.</td>
</tr>
<tr>
<td>USER_ERROR</td>
<td>Invalid Attachment record combination</td>
</tr>
<tr>
<td>USER_ERROR</td>
<td>Missing Item Weight or Weight Unit.</td>
</tr>
<tr>
<td>USER_ERROR</td>
<td>Missing or Invalid RecordType for AttachTo</td>
</tr>
<tr>
<td>USER_ERROR</td>
<td>Must submit a non-abstract instance of baseRef (eg RecordRef, CustomRecordRef) NOT a baseRef</td>
</tr>
<tr>
<td>USER_ERROR</td>
<td>Must submit a non-abstract instance of record or searchRecord (eg customer or customerSearchBasic).</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>USER_ERROR</td>
<td>No budget inserted or updated.</td>
</tr>
<tr>
<td>USER_ERROR</td>
<td>{1}</td>
</tr>
<tr>
<td>USER_ERROR</td>
<td>Please enter value(s) for: Company Name, Subsidiary</td>
</tr>
<tr>
<td>USPS_ACCT_NUM_ALREADY_EXISTS</td>
<td>There is an existing NetSuite registration for Endicia account number {1}.</td>
</tr>
<tr>
<td>USPS_INVALID_INSURED_VALUE</td>
<td>Insured value exceeds the {1} maximum allowed by Endicia.</td>
</tr>
<tr>
<td>USPS_INVALID_PACKAGING</td>
<td>The Carrier Packaging that you have selected is not valid for this item fulfillment. &lt;br&gt; Usually this indicates the selected packaging cannot be used with the selected USPS shipping method, or the package weight is invalid. &lt;br&gt; Please check the documentation for more details.</td>
</tr>
<tr>
<td>USPS_INVALID_PSWD</td>
<td>The Endicia Web Password does not match the Web Password for this USPS Registration account number.</td>
</tr>
<tr>
<td>USPS_LABEL_VOIDED</td>
<td>This error required 1 or more labels created for this transaction to be voided at Endicia.</td>
</tr>
<tr>
<td>USPS_LABEL_VOIDED</td>
<td>This error required 1 or more labels created for this transaction to be voided at Endicia.&lt;br&gt;</td>
</tr>
<tr>
<td>USPS_MAX_ITEM_EXCEEDED</td>
<td>International USPS fulfillments allow a maximum of 5 unique items per package, due to customs documentation. If more than one package is required, please break up the shipment into multiple fulfillments.</td>
</tr>
<tr>
<td>USPS_ONE_PACKAGE_ALLWD</td>
<td>International USPS fulfillments allow only one package. If more than one package is required, please break up the shipment into multiple fulfillments of one package each.</td>
</tr>
<tr>
<td>USPS_PASS_PHRASE_NOT_UPDATED</td>
<td>The Endica Pass Phrase was not updated: {1}</td>
</tr>
<tr>
<td>USPS_REFUND_FAILED</td>
<td>Failed Endicia Refund Request</td>
</tr>
<tr>
<td>USPS_REFUND_FAILED</td>
<td>The Endicia Refund Request failed due to a system error.</td>
</tr>
<tr>
<td>USPS_RETRY</td>
<td>A response was not received for the USPS funding request. Please try again in a few minutes.</td>
</tr>
<tr>
<td>USPS_VALIDATE_ADDR</td>
<td>The address you entered could not be validated. Please verify the city, state, and/or zip code.&lt;br&gt; You can validate an address by visiting the &lt;a href=&quot;http://zip4.usps.com/zip4/welcome.jsp&quot; target=&quot;_blank&quot;&gt;U.S. Postal Service&lt;/a&gt; web site, or the &lt;a href=&quot;http://www.endicia.com/Developers/ZipLookup/&quot; target=&quot;_blank&quot;&gt;Endicia&lt;/a&gt; web site.</td>
</tr>
<tr>
<td>USPS_VERIFY_TRACKING_NUM</td>
<td>Please verify that the following tracking numbers were created and voided in your Endicia account before proceeding.</td>
</tr>
<tr>
<td>USPS_VOID_ERROR</td>
<td>An error was detected during the Endicia Void operation:</td>
</tr>
<tr>
<td>USPS VOID_ERROR</td>
<td>An error was detected during the Endicia Void operation:&lt;br&gt;</td>
</tr>
<tr>
<td>VALID_EMAIL_REQD</td>
<td>Missing or invalid email address. Email address is a required field and it cannot be null or empty. The email address must be in a valid format.</td>
</tr>
<tr>
<td>VALID_EMAIL_REQD_FOR_LOGIN</td>
<td>Please enter a valid email address when granting login access privileges to this record.</td>
</tr>
<tr>
<td>VALID_FIRST_NAME_REQD</td>
<td>Missing or invalid First Name. Users first name is a required field and cannot be null or empty.</td>
</tr>
<tr>
<td>VALID_LAST_NAME_REQD</td>
<td>Missing or invalid Last Name. Users last name is a required field and cannot be null or empty.</td>
</tr>
<tr>
<td>Error Code Returned</td>
<td>Long Description or Message</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>VALID_LINE_ITEM_REQD</td>
<td>You must have at least one valid line item for this transaction.</td>
</tr>
<tr>
<td>VALID_PHONE_NUM_REQD</td>
<td>Missing or invalid Home phone number. The Home phone number is a required field and it cannot be null or empty. The format of the Home phone number must contain area code plus seven digit number.</td>
</tr>
<tr>
<td>VALID_PRD_REQD</td>
<td>Insert Transaction Failure: No valid, open, posting period for date - {1}. Please visit Setup &gt; Manage Accounting Periods to set up a new accounting period.</td>
</tr>
<tr>
<td>VALID_PRD_REQD</td>
<td>Insert Transaction Failure: No valid, open, tax period for date - {1}. Please visit Setup &gt; Manage Tax Periods to set up a new tax period.</td>
</tr>
<tr>
<td>VALID_PRD_REQD</td>
<td>Update Transaction Failure: No valid, open, {1} period for date - {2}</td>
</tr>
<tr>
<td>VALID_URL_REQD</td>
<td>Please go back and provide a valid URL for all five fields on the External tab.</td>
</tr>
<tr>
<td>VALID_VERSION_REQD_IN_URL</td>
<td>If the version parameter is passed through the URL, it MUST contain a valid version in a phased release environment. Valid: (1)</td>
</tr>
<tr>
<td>VALID_WORK_PHONE_REQD</td>
<td>Missing or invalid Work phone number. The Work phone number is a required field and it cannot be null or empty. The format of the Work phone number must contain area code plus seven digit number.</td>
</tr>
<tr>
<td>VALID_ZIPCODE_REQD</td>
<td>Missing or invalid ZIP code field. ZIP code is a required field and it cannot be null or empty. ZIP code and state values are checked against an internal database to make sure that ZIP code specified exists in state specified.</td>
</tr>
<tr>
<td>VENDOR_TYPE_REQD</td>
<td>No Vendor Type was specified. If creating a Tax Agency, please ensure that the vendor type is active and marked as a tax agency.</td>
</tr>
<tr>
<td>VERIFY_DESTNTN.Zip_CODE</td>
<td>Please verify that the destination zipcode is correctly specified.</td>
</tr>
<tr>
<td>VERIFY_PAYROLL_FUND_ACCT</td>
<td>The payroll funding account has not been verified. Please verify the payroll funding account</td>
</tr>
<tr>
<td>VERIFY_ZIP_CODE_SETUP</td>
<td>Please verify that you have correctly set your zip code under setup company. If you have multi-location enabled, verify that you have set a correct zipcode for each location.</td>
</tr>
<tr>
<td>VISA_ERROR</td>
<td>Communication error with Visa. Please retry.</td>
</tr>
<tr>
<td>VOID_FAILED</td>
<td>Void Failed. Failed to void transaction, retry void or issue credit.</td>
</tr>
<tr>
<td>VOIDING_REVERSAL_DISALLOWED</td>
<td>You may not create a voiding reversal for transactions with inventory impact. To reverse the inventory impact of the transaction, you will need to create an inventory adjustment.</td>
</tr>
<tr>
<td>VSOE_CANT_ADD_ITEM_GROUP</td>
<td>When the &lt;b&gt;Is VSOE bundle&lt;/b&gt; box is checked, Items for Purchase cannot be added to item groups.</td>
</tr>
<tr>
<td>VSOE_REV_REC_TMPLT_REQD</td>
<td>All Lines in a VSOE Bundle with a VSOE Allocation must have a revenue recognition template.</td>
</tr>
<tr>
<td>VSOE_TOTAL_ALLOCATION_ERROR</td>
<td>The total vsoe allocation in a bundle must equal the total bundle sales amount.</td>
</tr>
<tr>
<td>VSOE_TRAN.VSOE_BUNDLE_ERROR</td>
<td>You have indicated that you would like this transaction to be treated as a Bundle (multi-element arrangement) for VSOE purposes. Please either uncheck the 'Transaction Is VSOE Bundle' checkbox or remove the Item Groups that have the 'Is VSOE Bundle' option specified.</td>
</tr>
<tr>
<td>WF_EXEC_USAGE_LIMIT_EXCEEDED</td>
<td>Workflow Execution Usage Limit Exceeded</td>
</tr>
<tr>
<td>WORK_DAYS_REQD</td>
<td>Select one or more working days.</td>
</tr>
<tr>
<td>WORLDPAY_ERROR</td>
<td>A failure occurred while attempting to connect to WorldPay.</td>
</tr>
</tbody>
</table>
### Error Status Codes

<table>
<thead>
<tr>
<th>Error Code Returned</th>
<th>Long Description or Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORLDPAY_ERROR</td>
<td>There was a problem with your WorldPay credentials. Please be sure you are using the</td>
</tr>
<tr>
<td></td>
<td>correct combination of Installation ID, Merchant Code and XML Password</td>
</tr>
<tr>
<td>WRITE_OFF_ACCT_REQD</td>
<td>To receive items without restocking, you must first set a value for the write-off account.</td>
</tr>
<tr>
<td></td>
<td>To set the value of the write-off account, go to Accounting &gt; Accounting Preferences &gt;</td>
</tr>
<tr>
<td></td>
<td>Order Management &gt; Write-Off Account for Returns.</td>
</tr>
<tr>
<td>WS_CONCUR_SESSION_DISALLWD</td>
<td>Someone has logged in as this user from a different SOAP web services session. Only one</td>
</tr>
<tr>
<td></td>
<td>person may login as a particular user at a time. As a consequence, this session has been</td>
</tr>
<tr>
<td></td>
<td>terminated.</td>
</tr>
<tr>
<td>WS_EXCEEDED_CONCUR_USERS_ALLWD</td>
<td>You can have a maximum of {1} active concurrent WS users at a time in {2}</td>
</tr>
<tr>
<td>WS_FEATURE_REQD</td>
<td>You have not enabled SOAP web services feature for your account.</td>
</tr>
<tr>
<td>WS_INVALID_SEARCH_OPERATN</td>
<td>When using request-level credentials, you must use the {1} operation instead of {2}</td>
</tr>
<tr>
<td>WS_LOG_IN_REQD</td>
<td>You must log in before performing a web service operation.</td>
</tr>
<tr>
<td>WS_PERMISSION_REQD</td>
<td>You do not have permission to access SOAP web services feature.</td>
</tr>
<tr>
<td>WS_REQUEST_BLOCKED</td>
<td>SuiteTalk concurrent request limit exceeded. Request blocked.</td>
</tr>
<tr>
<td>ZIP_FILE_CONTAINS_VIRUS</td>
<td>The zip file contains a virus {1}. Upload abort.</td>
</tr>
</tbody>
</table>

### Warning Status Codes

The code USER_ERROR is returned in the code field of the statusDetail type where the type attribute has a value of warning. A warning message is also returned which varies depending on the exact cause of the warning and is usually self-descriptive. For a complete description of warnings and how they differ from errors and faults, refer to [SOAP Web Services Warnings, Errors, and Faults](#).
Frequently Asked Questions

SOAP Web Services FAQ

Where is the NetSuite WSDL located?

Use the following URL to access the current SOAP WSDL:

https://webservices.netsuite.com/wsdl/v2019_2_0/netsuite.wsdl

(where v2019_2_0 is the version of the WSDL).

You can also use a URL of this type to access previous WSDLs. For example, the SOAP 2015.2 WSDL is located at:

https://webservices.netsuite.com/wsdl/v2015_2_0/netsuite.wsdl

How does the host data center for an account affect SOAP web services access?

The URL to be used for SOAP web services access varies depending upon the data center hosting each NetSuite account.

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

For more details about dynamic discovery of URLs, see Effects of Accounts' Host Data Center on URLs and Dynamic Discovery of URLs for SOAP Web Services and RESTlet Clients.

As of 2017.2, account-specific domains are supported for SOAP web services, and you can access your SOAP web services domain at the following URL, where 123456 is your account ID:

123456.suitetalk.api.netsuite.com. The data center-specific domains supported before 2017.2 will continue to be supported. For more information, see the help topic URLs for Account-Specific Domains.

What programming languages and platforms are supported in this release?

This release should work with any platform or tool that supports SOAP 1.1. NetSuite has verified that the Java Apache Axis and Microsoft .NET platforms work correctly, and sample programs for these environments are available in the Resources section of the Web Services portal. Also note that there is a PHP toolkit available for PHP developers wanting to build SOAP web services applications. See SOAP Web Services PHP Toolkit for more details.

What is an internal ID (or nsKey)? What is a RecordRef? How do I get the values for internal IDs?

An internal ID is a unique key used to reference a record within NetSuite. In SOAP web services, the RecordRef type is used to abstract the internal ID. For example, a RecordRef is required to uniquely identify the records for update, delete, or get operations. Also, a typical record exposed through SOAP web services will have references to other NetSuite records of different types through the RecordRef type.

The internal ID for a record can be obtained as follows:

- If a record type has been exposed through SOAP web services, a search operation will return the internal ID.
Alternatively, you can also obtain the internal ID for that record in the UI on the list page for that record type. In order to see the internal ID, go to Home > Set Preferences > General and select the Show Internal IDs check box.

What are the maximum number of records allowed for write operations such as addList, updateList, or deleteList?

This varies depending on whether records are submitted during peak or off-peak hours. For more detailed information, see the section called Understanding Record Limiting in the NetSuite Help Center.

Why can’t I maintain a valid session using Apache Axis?

The NetSuite SOAP web services implementation requires the client application to support multiple cookies on one line, as is the standard for cookies. There is a bug in Apache Axis that puts each cookie on its own line in the HTTP Headers. A patched version of the axis.jar can be downloaded from NetSuite Web Services Developer's page to fix this problem. Please note that the patch you download depends on the version of Axis that you are using. Replace the existing axis.jar file in the lib directory of your project with the NetSuite version.

Why can’t I maintain a valid session using Microsoft .NET?

The NetSuite SOAP web services implementation requires the client application to support multiple cookies. Please see the .NET sample application for an example of how to enable support for multiple cookies.

I have populated some fields in a record when adding or updating, but why are these fields not being populated in the SOAP document?

This is a .NET anomaly. For every field that’s not of type string, the generated .NET object contains an additional attribute whose name is the field name concatenated with the word "Specified". For example, there is a Boolean field name called inactive in the Customer record type. When .NET generates this class, it adds an additional Boolean field called inactiveSpecified. This field needs to be set to true for the inactive field to be included in the SOAP document.

Can multiple users logged into the same account run SOAP web services simultaneously?

Yes. SOAP web services are associated to the user, NOT the account. Therefore, multiple users can access the same account via SOAP web services. Also, a single user can have a concurrent browser and SOAP web services session, and additional SOAP web services sessions for a certain user can exist for SOAP web services partner applications.

Is it possible to run SOAP web services and XML independently and simultaneously within an account without mutually exclusive restrictions?

There are currently no restrictions. However, to ensure adequate load distribution, the use of all NetSuite integration technologies will be analyzed and restrictions implemented as appropriate.

How do I query for all records of a certain type?

Use the search operation with a search record that has no fields populated.

I get a SOAP fault and a message which states “The Operation has timed-out.” when using my .Net client and I get no SOAP response back. What could be causing this?

If you have not explicitly set the timeout on your .Net application, this may be related to the preset timeout of 100 seconds on the SoapHttpClientProtocol class. You must change this timeout to be
equal to or greater than the NetSuite timeout (15 minutes). Otherwise, your client will abort processing prematurely.

**Note:** If you receive the same fault, or a service unavailable error which includes an HTTP error code (for example, 50x), this is likely a NetSuite related timeout. Server restarts can occur, and if you receive the HTTP 503 error code, you need to retry sending the request later.

Sample Code: .Net

```csharp
public NSClient() {
    _isAuthenticated = false;
    _pageSize = 40;
    // Instantiate the NetSuite web services
    _service = new NetSuiteService();
    // Set client timeout to 100 hours
    _service.timeout = 100*60*60*1000;
}
```

Sample Code: Java

```java
// Locate the NetSuite web service.
NetSuiteServiceLocator service = new NetSuiteServiceLocator();
// Enable client cookie management. This is required.
service.setMaintainSession(true);
// Get the service port
_port = service.getNetSuitePort();
// Setting client timeout to 2 hours for long running operations
((NetSuiteBindingStub) _port).setTimeout(1000 * 60 * 60 * 2);
```

I am getting an unexpected server side protocol timeout error (502) and no SOAP response is sent back. What could be causing this?

Check the timeout settings on any intermediaries that you are using — such as proxy servers or port sniffers (SOAPScope).

To edit the timeout setting for SOAPScope, you need to edit the `httpd.conf` file to the desired settings and then restart the SOAPScope service. The `httpd.conf` file is located in the conf subdirectory of the SOAPScope data directory.

**Note:** There is a link to the SOAPScope data directory in the Start menu.

In the following code the default Timeout value of 300 has been changed to 901.

```
PidFile nul:
    Timeout 901
    KeepAlive On
    MaxKeepAliveRequests 0
    KeepAliveTimeout 15
```

Why do I receive socket timeout exception errors?

Networks are inherently unreliable, and a SocketTimeoutException can occur anytime. To handle this error, you must implement a mechanism to keep sending your SOAP web services requests. For information about setting up a retry mechanism, see Web Services and RESTlet Concurrency Governance.
Why are attachments uploaded with Messages getting corrupted?

This may be due to incorrectly encoding a file before sending it. 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.

**Sample Code:**

- **Correct**

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

- **Incorrect**

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

How can I determine internal ID values for individual items in a custom list?

Generally, a certain list displays the internal IDs as one of the columns. For example, List > Relationships > Customers displays the internal IDs as the second column (when the Show Internal ID preference is 
enabled). In the NetSuite Help Center, see [Enabling the Show Internal IDs Preference](#) for more details.

When referencing items in a custom list, you must first reference the **SelectCustomFieldRef** by its 
internal ID, which is displayed in the respective custom field list (for example, for a custom customer field it would be displayed at Customization > Lists, Records, & Fields > Entity Fields).

Next, reference the internal ID to the Custom List or Record. Internal IDs to custom lists are NOT 
displayed in the UI. You must refer to the HTML source for the internal ID. When in the HTML source, 
search for the name of the value and note the integer which follows. For example, if referring to the **Items of Interest** custom list (provided in most test accounts) and need to select **Monitors** as one of the values, 
view the HTML source in the browser and find that string:

```
```

**Note:** The display order is in the order the values appear, but the internal IDs may be different. 
Monitors in this case has a internal ID of 2.

If you are referring to a Record, you need to locate the internal ID displayed in the column of the 
appropriate list.

**When working with custom records, why do I need a separate CustomRecordRef class, 
when even for creating a customRecord I have to instantiate the RecordRef type?**

In general RecordRef is for **get()** and Record is for **add()**. We sometimes use RecordRef to store name/ 
value pairs.

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 can NOT get a CustomRecord by setting the RecordRef.type to customRecord on add — you must use CustomRecordRef to specify the kind of CustomRecord.

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.

Java Example

To add a CustomRecord

```java
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 type
    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 CustomRecord:

```java
CustomRecordRef myCRR = new CustomRecordRef();
    myCRR.setKey("1200"); // The key we got back
    myCRR.setTypeId("3"); // For Authors
    myCRR.setType(RecordType.customRecord);
    ReadResponse rr = port.get(myCRR);
```

.Net Example

To add a CustomRecord

```csharp
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 CustomRecord:

```csharp
CustomRecordRef myCRR = new CustomRecordRef();
    myCRR.internalId = "1200"; // The key we got back
    myCRR.typeId = "3"; // For Authors
    ReadResponse rr = _service.get(myCRR);
```

How can I set multiple fields' values to NULL when using the PHP toolkit?

You can set values to NULL by submitting the fields in nullFieldList. The following example sets the fax and phone field to NULL.
// Include NetSuite Class Library
require_once '../PHPToolkit/NetSuiteService.php';

// Instantiate new service
$service = new NetSuiteService();

// Create contact record
$ctact = new Contact();

// Set up record fields
$aBook = array (
    "addressbook" => array( 
        array( 
            "defaultBilling" => true, 
            "defaultShipping" => false, 
            "add1" => $_POST['bill_add1'], 
            "city" => $_POST['bill_city'], 
            "state" => $_POST['bill_state'], 
            "zip" => $_POST['bill_zip'] ),
        array( 
            "defaultBilling" => false, 
            "defaultShipping" => true, 
            "add1" => $_POST['ship_add1'], 
            "city" => $_POST['ship_city'], 
            "state" => $_POST['ship_state'], 
            "zip" => $_POST['ship_zip'] )
    ) );

// Use setFields to fill the contact record with the fields
setFields($ctact, array("addressbookList" => $aBook));

// Create NullField record
$nfl = new NullField();

// Set the null fields in the record
setFields($nfl, array("name" => array("fax", "phone"), 
    
setFields($ctact, array("internalId" => "87", "nullFieldList" => $nfl));

// Update the contact record
$response = $service->update($ctact);

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

Why do I not get data for all elements on a search for a certain record type?
When performing searches on records, by default only the body fields are returned.
For example, if a Get or GetList operation against a customer record is submitted, the addressbooklist element contains data but if a Search operation against a customer record is submitted, the addressbooklist element is un-defined.
To return field values for all lists on a record, you must set the bodyFieldsOnly element of the searchPreferences type to false. It is recommended that the default settings are used wherever list values are not necessary since this significantly improves performance.
<complexType name="SearchPreferences">
   <sequence>
      <element name="bodyFieldsOnly" type="xsd:boolean" default="true" />
   </sequence>
</complexType>

<element name="searchPreferences" type="platformCore: SearchPreferences" />

**Note:** You can perform quick and full searches as the same time. For example, to show a list of email folders, but only query for messages in the default folder, do a quick search to get folder names/keys, and a full search on all messages.

Does SOAP web services support double-byte character sets like those used in Japanese and Korean?

The WSDL itself is only in English, but data can be in any language that uses UTF-8 character encoding, including those that use double-byte character sets.

Why are some classes from the NetSuite WSDL not being generated?

If you are using Java, to generate unreferenced types you should set the **all** parameter in your axis-wsdl2java ant task to **true**. For example:

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

What is the process flow for using SOAP web services asynchronous calls?

The process flow for using WS asynchronous calls is as follows:

1. Make the desired asynchronous call defined in NetSuitePortType – asyncAddList, asyncUpdateList, asyncDeleteList, asyncGetList, asyncSearch. A job ID is returned.
2. Call the checkAsyncStatus method with the job ID as the parameter. The returned object, AsyncStatusResult, encapsulates the object AsyncStatusType, which indicates the status of the asynchronous job. The defined statuses are failed, finishedWithErrors, pending, processing, and finished. This step may iterate until the desired status is reached. Alternatively, navigate to Setup > Integration > SOAP Web Services Process Status to obtain the status information.
3. Call the getAsyncResult method with the job ID and page index as the parameters. The returned object AsyncResult contains the result of each record on which the asynchronous job operated.

**Note:** AsyncResult is an abstract type; hence one of its concrete subclasses AsyncAddListResult, AsyncDeleteListResult, AsyncGetListResult, AsyncSearchResult, AsyncUpdateListResult – is returned.

Are SOAP web services samples available for different development platforms?

The NetSuite SOAP web services implementation is designed to work with any development platform that is able to generate SOAP messages. We provide Java and C# sample applications, which are available for download in our SuiteTalk portal (http://www.netsuite.com/portal/developers/resources/suitetalk.shtml).

You may also find useful community-based developer assistance at the web services forum in the NetSuite Usergroup.
Can I use SOAP web services with my sandbox account?

Please note that the URL for the WSDL differs between your production account and your sandbox account. If you want to use SOAP web services with your sandbox account you will need to use the following URL:

https://<accountID>-sb1.suitetalk.netsuite.com

**Note:** Provide your own account ID in your SOAP web services request.

Where can I download the current version of the WSDL?

The NetSuite SOAP WSDL provides a complete description of available services, operations, and messages for SOAP web services.

**To download the current version of the WSDL:**

2. Click View Zipped.

For additional information, see the following areas in the NetSuite Help Center:

- **Release Notes** – Describes all changes in the newest release, including an overview of platform enhancements and a list of newly exposed records.
- **Platform Guide** – Offers an overview of the NetSuite SOAP web services architecture. Includes a “Getting Started” section for users of both C# and Java.
- **Records Guide** – Features detailed reference pages on all supported records.

For downloadable versions of Help Center documentation, see the help topic [User Guides](#).