



SuiteTalk (Web Services) Platform Guide

Version: 2007.1
Rev Date: December 12, 2007

Copyright NetSuite, Inc. 2005 All rights reserved.

SuiteTalk Platform - Version 2.6.0

December 12, 2007

This document is the property of NetSuite, Inc., and may not be reproduced in whole or in part without prior written approval of NetSuite, Inc.

Trademarks

NetSuite, NetERP and NetCRM are provided by NetSuite, Inc, and NetSuite is a trademark of NetSuite, Inc.

Oracle is a registered trademark of Oracle Corporation.

Other product names mentioned in this document may be trademarks, servicemarks, or tradenames of their respective companies and are hereby acknowledged.

Contents

Chapter 1

SuiteTalk Platform Overview

In this Guide	8
Supported Operations	8
<i>List Operations</i>	9
<i>Asynchronous Operations</i>	9
Understanding Web Services Governance	11
<i>Understanding Record Limiting</i>	12
<i>Understanding Request Limiting</i>	13
<i>Understanding Governance Errors</i>	13
Creating Integration Reports	13
Vocabulary	14

Chapter 2

Getting Started

Understanding Web Services	15
Quick Start	16
<i>Enabling the Web Services Feature</i>	16
<i>Building an Application with Microsoft .NET</i>	16
<i>Building an Application with Java using Apache Axis</i>	18
<i>Running a Sample Application</i>	20
<i>Using Web Services with PHP</i>	20
<i>Using SOAPScope</i>	21
Setting Your Web Services Preferences	22
<i>Setting Company-Wide Preferences</i>	22
<i>Resetting Default Behavior</i>	24
<i>Setting Preferences at the Request Level</i>	25
<i>Displaying Internal IDs</i>	27
Understanding NetSuite Roles and Permissions	27
<i>Setting a Default Role for a Web Services User</i>	28
<i>Setting a Web Services Only Role for a User</i>	28
<i>Customer Center, Vendor Center, and Partner Center Roles</i>	29
<i>IDs Associated with Roles</i>	30
General Concepts	31
<i>Understanding NetSuite Features</i>	31
<i>PCI Compliance Password Requirements</i>	31
<i>Using Web Services Specific Forms</i>	32
<i>Effects of Account Configuration on Web Services</i>	32
<i>Working with Sublists</i>	32
<i>Working with Hidden Fields</i>	35

Chapter 3

Platform Features

Understanding the WSDL and XSD Structure	37
<i>Versioning</i>	38
<i>NetSuite Messaging XSD Files</i>	38
<i>NetSuite Business Records XSD Files</i>	38
<i>System Constants XSD Files</i>	39
<i>Example</i>	40
Security	41
<i>Authentication</i>	41
<i>Authorization</i>	42
<i>Session Management</i>	42
<i>Encryption</i>	45
Synchronous and Asynchronous Request Processing	45
<i>checkAsyncStatus</i>	46
<i>getAsyncResult</i>	47
<i>Monitoring Asynchronous Jobs in the UI</i>	48
Using Internal IDs, External IDs, and References	48
<i>Understanding ExternalIDs</i>	50
<i>SuiteTalk Operations and Their Internal ID Requirements</i>	50
<i>Example Internal ID Usage</i>	51
Web Services to UI Single Login	55
Handling Errors	56
<i>Warnings</i>	56
<i>Errors</i>	57
<i>Faults</i>	58

Chapter 4

Types

Built-in Types	59
Complex Types	60
<i>Passport</i>	60
<i>Record</i>	61
<i>RecordList</i>	61
<i>BaseRef</i>	61
<i>RecordRef</i>	62
<i>CustomRecordRef</i>	62
<i>ListOrRecordRef</i>	62
<i>Status</i>	63
<i>StatusDetail</i>	63
<i>NullField</i>	63
<i>ReadResponse</i>	64
<i>ListReadResponse</i>	64
<i>ListWriteResponse</i>	64
<i>WriteResponse</i>	64
Custom Field Types	65
<i>CustomFieldRef</i>	65
<i>IntCustomFieldRef</i>	65
<i>DoubleCustomFieldRef</i>	66

<i>BooleanCustomFieldRef</i>	66
<i>StringCustomFieldRef</i>	66
<i>DateCustomFieldRef</i>	66
<i>SelectCustomFieldRef</i>	66
<i>MultiSelectCustomFieldRef</i>	67
<i>CustomFieldList</i>	67
Search Types	67
<i>Search XML Schema Types</i>	67
<i>Search Custom Field XML Schema Types</i>	70
<i>Sample Code</i>	73
Platform Enumerations	74

Chapter 5

Operations

login	79
<i>Request</i>	79
<i>Response</i>	80
<i>Faults</i>	80
<i>Sample Code</i>	81
mapSso	84
<i>Request</i>	84
<i>Response</i>	85
<i>Faults</i>	85
<i>Sample Code: SOAP</i>	85
<i>Sample Code: Java</i>	85
logout	86
<i>Request</i>	86
<i>Response</i>	86
<i>Faults</i>	86
<i>Sample Code</i>	86
addList	87
<i>Request</i>	88
<i>Response</i>	88
<i>Faults</i>	88
<i>Sample Code</i>	88
add	89
<i>Request</i>	89
<i>Response</i>	89
<i>Faults</i>	89
<i>Sample Code</i>	90
updateList	94
<i>Request</i>	94
<i>Response</i>	94
<i>Faults</i>	95
<i>Sample Code</i>	95
update	98
<i>Request</i>	98
<i>Response</i>	98
<i>Faults</i>	99

<i>Sample Code</i>	99
<i>Updating Record Lists</i>	101
deleteList	101
<i>Request</i>	101
<i>Response</i>	102
<i>Faults</i>	102
<i>Sample Code</i>	102
delete	105
<i>Request</i>	105
<i>Response</i>	105
<i>Faults</i>	105
<i>Sample Code</i>	105
search	107
<i>Working with Searches and Joined Searches</i>	107
<i>Returning Associated Lists</i>	110
<i>Search Preferences</i>	111
<i>Filtering Lists that Contain Null Values</i>	112
<i>Request</i>	112
<i>Response</i>	113
<i>Faults</i>	113
<i>Sample Code</i>	113
searchMore	118
<i>Request</i>	119
<i>Response</i>	119
<i>Faults</i>	120
<i>Sample Code</i>	120
searchNext	122
<i>Request</i>	122
<i>Response</i>	122
<i>Faults</i>	123
<i>Sample Code</i>	123
getList	124
<i>Request</i>	124
<i>Response</i>	124
<i>Faults</i>	124
<i>Sample Code</i>	124
get	126
<i>Request</i>	126
<i>Response</i>	126
<i>Faults</i>	126
<i>Sample Code</i>	127
getAll	129
<i>Request</i>	129
<i>Response</i>	129
<i>Faults</i>	129
<i>Sample Code</i>	130
getSelectValue	134
<i>Request</i>	134
<i>Response</i>	134
<i>Faults</i>	135

<i>Sample Code</i>	135
<i>GetSelectValue Types</i>	136
getCustomization	136
<i>Request</i>	136
<i>Response</i>	137
<i>Sample Code</i>	137
getItemAvailability	138
<i>Request</i>	138
<i>Response</i>	138
<i>Faults</i>	139
<i>Sample Code</i>	139
attach / detach	140
<i>Request (attach)</i>	142
<i>Request (detach)</i>	142
<i>Response (attach)</i>	142
<i>Response (detach)</i>	142
<i>Faults</i>	142
<i>Sample Code</i>	142
changePasswordOrEmail	144
getDeleted	144
<i>Request</i>	145
<i>Response</i>	145
<i>Faults</i>	145
<i>Sample Code</i>	145
initialize / initializeList	146
<i>Sample Code</i>	148

Handling Errors

Soap Faults for Each Operation	150
Soap Fault Status Codes	152
Error Status Codes	153
Warning Status Codes	174

Task IDs

Chapter 1 SuiteTalk Platform Overview

The SuiteTalk Platform provides programmatic access to your NetSuite data and business processes through an XML-based application programming interface (API). This material pertains to the SuiteTalk 2.6 WSDL, and provides an overview of the SuiteTalk Platform with detailed requirements and example code for each of the core Web services operations. For detailed information on each record available, refer to the **SuiteTalk (Web Services) Records Guide**.

In this Guide

This manual contains the following sections:

Chapter 1 "SuiteTalk Platform Overview": provides an overview of this guide, what operations are supported, and an overview of Web services governance.

Chapter 2 "Getting Started": provides an overview of Web services and how to quickly get your web services development environment set up to create your first service. It also describes how to modify the default Web services preferences and provides some general information useful when working with many different operations or record types.

Chapter 3 "Platform Features": describes basic features of the NetSuite Web Services Platform including the WSDL and XSD structure, session management and error handling.

Chapter 4 "Types": describes the various types available in the SuiteTalk Platform.

Chapter 5 "Operations": describes each operation that can be performed through web services and provides SOAP, C# and Java code samples.

Handling Errors: provides tables of possible SOAP faults, fault status codes and error status codes that can be thrown for a given request.

Supported Operations

In this version of the SuiteTalk Platform the following user actions or operations are supported:

- **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 an account number. All other Web Services operations require a successful login. Once successfully completed, a login creates a session that allows subsequent operations to be performed without having to login again until the session expires or the logout operation is invoked. For more information on how sessions are controlled, refer to ["Session Management"](#) on page 42.
- **mapSso**: Use to automate the mapping between external applications credentials and NetSuite's credentials for a user.

- **logout**: Use to logout from the system. The logout operation invalidates the current session.
- **add / addList**: Use to add one or more records into the system. The system returns a NetSuite identifier (internalId) that is unique for each record created within a record type.
- **update / updateList**: Use to update one or more existing records in the system by providing new values for the fields to be updated for each record. The records to be updated are identified by either the internal or external ID and the record type.
- **delete / deleteList**: Use to delete one or more records in the system. The records to be deleted are identified by either the internal or external ID and the record type.
- **get / getList**: Use to query the system for one or more records. You must provide either the internal or external ID and the record type for each query item.
- **getAll**: Use to return the a list of records that do not have a search interface.
- **search / searchNext / searchMore**: 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 “Setting Your Web Services Preferences” on page 22.
- **getSelectValue**: Use to retrieve valid values for a given recordRef field where the referenced record type is not yet exposed in the Web services API or when the logged in role does not have permission to the instances of the record type.
- **getCustomization**: Use to dynamically retrieve and manage the metadata for Custom Fields, Lists, and Record Types
- **getItemAvailability**: Use to retrieve the inventory availability for a given list of items.
- **attach / detach**: Use to attach or detach another record or file to/from another record.
- **changePasswordOrEmail**: Use to change a users email or password.
- **getDeleted**: Use to retrieve a list of deleted records of a given type during a specified period.
- **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.

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, deleteList, getList, and initializeList) allows you to work with multiple record types. For example, with a single addList operation you can add 3 customers, 4 opportunities and 1 contact.

Asynchronous Operations

The following asynchronous equivalents are available for these list operations:

- addList: asyncAddList
- updateList: asyncUpdateList

- deleteList: asyncDeleteList
- getList: asyncGetList
- search: asyncSearch
- initializeList: asyncInitializeList

For information on asynchronous request processing, see [“Synchronous and Asynchronous Request Processing”](#) on page 45.

Understanding Web Services Governance

In order to optimize NetSuite application and database servers, we have implemented a number of mechanisms to control the consumption of Web services. These mechanisms ensure the following:

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

WS governance is made up of two primary areas:

- Record Limiting (see “[Understanding Record Limiting](#)” on page 12)
- Request Limiting (see “[Understanding Request Limiting](#)” on page 13)

Understanding Record Limiting

An individual request is rejected if it exceeds a certain size. The allowed record size varies depending on several factors, including the time of day the operation is submitted, and whether it is sent asynchronously or synchronously.



Important: The record limits provided in the following tables are on a **per request** basis. There is NO limit to the number of requests that can be sent within a given time period, only on the number of records sent in an individual request.

Synchronous Operations



Note: Peak hours are considered 6am to 6pm PST, Monday through Friday.

Peak Hours

Operation (on a per request basis)	Record Count
Add	100
Update	50
Delete	100
Search Page Size	500

Off-Peak Hours

Operation (on a per request basis)	Record Count
Add	200
Update	100
Delete	200
Search Page Size	1000

Asynchronous Operations

Record limits for asynchronous requests are the same regardless of whether they are sent during peak or off-peak hours.

Operation (on a per request basis)	Record Count
Add	400
Update	200
Delete	400
Search Page Size	2000

Understanding Request Limiting

Request limiting refers to the restrictions on the number of concurrent requests that can be made. The number of concurrent requests allowed varies depending on request size and time of day (peak or off-peak hours).

Request Types

Requests are classified as Long or Short depending on the number of records submitted in the request, as follows:

Request Type	# of Records (Add/Delete)	# of Records (Update)	
Long	100+	50+	> 300
Short	< 4	< 4	< 10

Request limiting is also governed by your License agreement, but in general, no more than 1 long or 3 short requests can occur concurrently on a single account.

Understanding Governance Errors

The following faults can be thrown as a result of other governance violations.

- **ExceededRecordCountFault:** fault thrown if a request exceeds the allowed record count (see “[Understanding Record Limiting](#)” on page 12)
- **ExceededRequestLimitFault:** fault thrown if the allowed number of concurrent requests is exceeded (see “[Understanding Request Limiting](#)” on page 13)

For more information on exceptions, refer to “[Handling Errors](#)” on page 56.

Creating Integration Reports

You can monitor Web services processing at any time by accessing Web Services Integration reports. Create your own Web services reports to obtain details on:

- Web services performance statistics
- Type of operations used (e.g. add, addList, delete, etc.)
- Number of records processed in an operation
- Who performed an operation
- Time an operation was performed
- Overall work done through Web services

Web Services Integration reports are also useful for administrators who need to diagnose and troubleshoot issues.



Important: Integration Reports permission is required to view these reports.

To access these reports, go to Reports > Integration and select either of the following:

- Integration and Automation Usage Summary By Job (**Note:** SOAP files are stored in the "View" links in the Request and Response columns.)
- Integration and Automation Usage Summary By Record Type

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 additional details, refer to the online help.

Vocabulary

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

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

Chapter 2 Getting Started

This section provides the basic information you need to get started using the SuiteTalk Platform and includes the following sections:

- [Understanding Web Services](#): provides a basic description of Web services.
- [Quick Start](#): provides step-by-step instructions on how to setup your Web services environment and start building applications to interface with the SuiteTalk Platform. Also includes a section that describes how to run a sample application provided by NetSuite.
- [Setting Your Web Services Preferences](#): describes how to adjust preferences specific to the SuiteTalk Platform.
- [General Concepts](#): provides information about the NetSuite data model that is useful to consider when building your own applications to interface with the SuiteTalk Platform.

Before building a Web services application to interface with the SuiteTalk Platform, the Web Services feature must first be enabled in the user interface (UI). To do this, go to Setup > Company > Enable Features. Click the Customization & Integration tab, select the Web Services check box, and click Save. Once you have enabled the feature, you can set your Web services preferences (see [“Setting Your Web Services Preferences”](#) on page 22).

Understanding Web Services

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

The following protocols are used to publish, expose or access Web Services:

- **WSDL** (Web Services Description Language): a WSDL file exposes a Web service interface, interaction patterns and protocol mapping. WSDL can be readily interpreted by other applications, systems and platforms.
- **UDDI** (Universal Description, Discovery and Integration): A Web Service can be categorized and registered in a UDDI Registry so that applications can locate it and retrieve its WSDL. Currently NetSuite does NOT provide a UDDI registry.

- **SOAP** (Simple Object Access Protocol): A Web Service can use the SOAP messaging format to define an envelope for Web services communication via HTTP, HTTPs or other transport layers.

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

Quick Start

This section provides details on how to use Microsoft .NET or Java to build an application that uses NetSuite Web services and how to run the sample application provided by NetSuite.

Enabling the Web Services Feature

The Web services feature **must** be enabled prior to submitting Web services requests.

To enable the Web services feature:

1. As administrator, click Setup > Company > Enable Features.
2. Click the Customization & Integration tab.
3. Select the Web Services check box.
4. Click Save.

Building an Application with Microsoft .NET

This section provides details on how to use the Microsoft .NET platform to build an application that uses NetSuite Web services.



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 version 1.1. 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 — Visual Basic, Visual J#, and C++.

To use Microsoft.NET with NetSuite Web services:

1. Install the Microsoft .NET framework.
Install Microsoft Visual Studio .NET, version 2003 or higher, which includes the .NET framework SDK 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:
 - a. Open Microsoft Visual Studio .NET.

- b. Create a new project by selecting the appropriate language and an application type.
- c. Once the project has been created, select the Add Web Reference option from the Project menu.
- d. When prompted, enter the NetSuite WSDL URL and then click Go.
Note: The URL to the beta version of the SuiteTalk 2.6 WSDL is:
`https://webservices.netsuite.com/wsd/v2_6_0/netsuite.wsdl`
The URL to the SuiteTalk 2.5 WSDL is:
`https://webservices.netsuite.com/wsd/v2_5_0/netsuite.wsdl`
Visual Studio inspects the WSDL and displays a summary of the available operations.
- e. Click Add Reference to generate the classes.
Once complete, you can view these classes in the Solution Explorer by expanding the Web Reference icon.
The generated proxy classes are available in a file named *Reference.cs*. In order to view this file, you may first need to enable the Show All Files option under the Project menu.

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

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

```
wSDL /language:<language>cs <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/wsd/v2_5_0/netsuite.wsdl
```

A C# file called `NetSuiteService.cs` is generated.
 - c. Compile the source code for your proxy into a .NET Assembly using the .NET Framework C# compiler or any other supported compiler.
 - d. Locate the `csc.exe` file under the Microsoft .NET Framework SDK installation and add it to your system path.

```
csc.exe /out: NetSuiteService.dll /target:library  
reference:system.xml.serialization.dll /  
reference:system.web.services.dll  
NetSuiteService.cs
```
3. Implement your application by writing your business logic using the generated .NET proxy classes.
Note that the following code snippets are shown in C#.
 - a. Instantiate the NetSuite service.

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

- b. Enable support for multiple cookie management.

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

- c. Create a valid session by populating the Passport object.

```
invoking the login operation.
Passport passport = new Passport();
passport.account = "TSTD96";
passport.email = "username@netsuite.com";
RecordRef role = new RecordRef();
role.id = "3";
passport.role = role;
passport.password = "foobar1";
Status status = service.login( passport ).status;
```

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

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

- e. Logout to invalidate the current session.

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

Building an Application with Java using Apache Axis

This section provides details on how to use the Apache Axis framework version 1.1 and higher to build an application that uses the NetSuite Web services.



Note: The Java based sample application available on the beta portal provides an in-depth look on how to build an application using Apache Axis and NetSuite Web services.

To use the Apache Axis framework with NetSuite Web services:

1. Install the Java 2 Platform.

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

2. Install Apache Axis.

- a. Download and install Apache Axis, version 1.1 or higher, from <http://ws.apache.org/axis/>.

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

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

- c. Once 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.

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

4. Configure Java to generate unreferenced types.

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

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

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

Using Apache Axis from the command line

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

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

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

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

```
>set
CP=%AXIS_HOME%\lib\axis.jar;%AXIS_HOME%\lib\jaxrpc.jar;%
AXIS_HOME%\lib\commons-discovery.jar;%AXIS_HOME%\lib\wsdl4j.jar;
%AXIS_HOME%\lib\saaj.jar;>java -cp %CP% org.apache.axis.wsdl.WSDL2Java
https://webservices.netsuite.com/wsdl/v2_5_0/netsuite.wsdl
```

Using the Apache Axis task in Apache Ant

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

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

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

```
ant generate.interfaces
```

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

a. Locate the NetSuite service.

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

b. Enable support for multiple cookie management.

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

- c. Get the NetSuite port.

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

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

```
passport.setEmail( "username@netsuite.com" );
passport.setPassword( "foobar1" );
role.setid( "3" );
passport.setRole( role );
passport.setAccount( "TSTDRV96" );
Status status = port.login( passport ).getStatus();
```

- e. 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 );
```

- f. Logout to invalidate the current session

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

Running a Sample Application

A sample C# based application is available on the [NetSuite Web services portal](#). This is a functional command line web services application that provides the ability to run several operations on a customer record. Before creating your own applications, it is recommended that you run this application and familiarize yourself with the associated code.

Using Web Services with PHP

Important: NetSuite has not tested using NetSuite Web Services with PHP. The following information is to provide general guidelines only.

When using NetSuite Web services with PHP, you can do one of the following:

- Call Web services directly through PHP.

This is the preferred option. PHP itself does not directly support SOAP-based Web services. However, this functionality is available through third-party libraries such as the following:

PEAR::SOAP: <http://pear.php.net/package/SOAP>

NuSOAP: <http://sourceforge.net/projects/nusoap/>

PHP-SOAP: <http://phpsoaptoolkit.sourceforge.net/phpsoap/>

For an overview of how to create a SOAP client in PHP using one of the above libraries, refer to the following article:

<http://www.onlamp.com/lpt/a/3968>

- Call Web services using Java through a PHP/Java bridge

The PHP/Java bridge is a PHP module which connects the PHP object system with the Java object system. The advantage to using this method is that the Java code that encapsulates the Web services calls has been tested previously by NetSuite.

More information regarding a PHP/Java bridge, refer to the following article:

<http://php-java-bridge.sourceforge.net/>

Using SOAPScope

Mindreef's SOAPScope utility allows you to capture and analyze SOAP request and response messages associated with an operation call. To use SOAPScope with a NetSuite Web services client, follow the instructions in this section to set up SOAPScope's SSL forwarding feature.

Mindreef's SOAPscope 4.1's SSL forwarding feature allows you to use SOAPscope to capture SOAP traffic over SSL/TLS/HTTPS, logging the messages in cleartext while still using encryption between the client and server.



Note: You can also use SOAPScope to manually invoke requests for a specific operation. When invoking requests from within SOAPScope, remember that you must first submit the login operation just as with any other web services application. Also, to deselect all optional elements for a record, hit **CTRL/Click** on any element to deselect it's siblings.

To set up SOAPScope:

1. Install SOAPscope 4.1

You can obtain a trial version of SOAPscope at:

<http://www.mindreef.com/order/eval.php?partner=NETSUITE02>

2. Configure SOAPScope.

To use the SSL Forwarding feature, you must configure SOAPscope to listen for SSL traffic on a specified port. When your client connects to that port, SOAPscope decrypts the data, logs it, re-encrypts it and forwards the data to and from the Web service. SOAPscope uses a self-signed SSL certificate to validate the forwarding socket. However, most client applications treat a self-signed certificate as untrusted, because it has not been signed by a trusted Certificate Authority, and will refuse the connection. Therefore your client app may need to be modified slightly to allow it to connect.

Modifying a .NET/C# client application:

- Add the following class to your code:

```
class OverrideCertificatePolicy : ICertificatePolicy
{
    public bool CheckValidationResult(ServicePoint srvPoint,
        System.Security.Cryptography.X509Certificates.X509Certificate certificate,
        WebRequest request, int certificateProblem)
    {
        return true;
    }
}
```

The class **OverrideCertificatePolicy** implements the **ICertificatePolicy** interface, which defines how SSL certificates are to be validated. This implementation of **CheckValidationResult** unconditionally validates any server certificate that is presented.

- Before you instantiate the service object in your code, add the following line:

```
ServicePointManager.CertificatePolicy = new OverrideCertificatePolicy();
```

This creates an instance of this class and assigns it as the global certificate policy for the application.

Modifying an Axis client application:

Add the following line of code in your client *before* the code that instantiates the service object.

```
System.setProperty("axis.socketSecureFactory",  
"org.apache.axis.components.net.SunFakeTrustSocketFactory");
```

This disables certificate checking by the Axis client.

3. Change your SOAP endpoint in the client code to point to port 8443 on your local machine which allows SOAPscope to intercept the message and capture it, and then forward to the original destination. So the client code should bind to the following endpoint (where `_2_5` is the namespace version):

```
https://localhost:8443/services/NetSuitePort_2_5
```

The following files will need to be modified:

- For Microsoft .NET: Reference.cs
 - For Java: com.netsuite.webservices.platform.NetsuiteServiceLocator.java
4. Use `ssconfig.exe` to tell SOAPscope to listen for HTTPS on port 8443, and forward messages to the original destination host and port. In order to do this, open up a command prompt and change directories to the SOAPscope bin directory that contains the `ssconfig.exe` file (should be in `C:\Program Files\Mindreef\SOAPscope 4.1\bin`). Type the following:

```
ssconfig -s 8443,webservices.netsuite.com:443
```

5. Restart SOAPscope.
6. Compile your Web Services client and run.

Setting Your Web Services Preferences

Before using Web services with your NetSuite account, ensure that the appropriate preferences are enabled as defined in the following sections. These preferences control how certain services are executed.

Setting Company-Wide Preferences

You can set the following company-wide preferences at Setup > Integration > Web Services Preferences. Changes to these preferences are propagated to every user within the company.

- **Search Page Size:** Determines the number of records returned for a given search. The default page size is 1000. This value must be greater than 5 and less than 1000. The page size entered here can be overridden at the individual request level. However, the following exception can be thrown:
 - **ExceededMaxRecordsFault:** the page size is set to greater than the record size limit and the actual request includes more than the maximum allowed records. The request fails.
- **Use Conditional Defaults on Add:** See “Resetting Default Behavior” on page 24 for more information.

- **Treat Warnings as Errors:** If enabled, warning messages generated by Netsuite are treated as errors causing an exception to be thrown that results in rejection of the request. For more information on the difference between errors and warnings, refer to “[Handling Errors](#)” on page 56.
- **Use Conditional Defaults on Update:** See “[Resetting Default Behavior](#)” on page 24 for more information.
- **Disable Mandatory Custom Field Validation:** If enabled, when data for a custom field is required for NetSuite UI submissions, it is NOT required when submitting a Web services request. If not enabled, an error is thrown whenever the data for a required custom field is not provided.

It is recommended that you enable this setting for cases where values for a required custom field may not be provided, such as when integrating with an application that does not include equivalent fields. If this setting is not enabled, and a request does not include data for a mandatory custom field, a `CSTM_FIELD_VALUE_REQD` error is returned. The error does not provide details on the data required.

- **disableSystemNotesForCustomFields:** When importing data from Custom Fields, you have the option to disable the creation of system notes during the import for those fields. Depending on the size of your import, this may significantly increase performance. This preference is available at the request level only. To enable the preference, submit the following in your SOAP Header (see “[Setting Preferences at the Request Level](#)” on page 25 for more details on setting request level preferences):

```
<platformMsgs:disableSystemNotesForCustomFields>true  
</platformMsgs:disableSystemNotesForCustomFields>
```

Important: System generated notes are used in NetSuite to track changes to a record including what action was taken, when the record was modified and the user that was responsible for the change. This is important for maintaining a complete audit trail. If you turn off system generated notes for custom fields, specific changes related to custom fields within the imported record are NOT recorded in NetSuite. All changes for standard fields are logged as usual. Therefore, if a custom field contains sensitive information that is critical for audit purposes, you should NOT disable system generated notes.

- **Disable Client SuiteScript:** When enabled, Client SuiteScript is not run during a Web services operation. The default setting is **TRUE**.

Important: If you have enabled this preference and are experiencing unexpected errors during a Web services operation on a form that has Client SuiteScript associated with it, disable Client SuiteScript and then run the operation again to verify if Client SuiteScript is the cause of the problem.

- **Disable Server SuiteScript:** If you are doing a historical import, it is recommended that you disable Server SuiteScript. If you are syncing live data or running a partner application (for example, Outlook Sync) it is recommended that you enable Server SuiteScript to ensure your business logic is run for your integrated application. Note that running Server SuiteScript will have a negative performance impact.

Important: To ensure that certain business logic is always executed for your integrated processes, use Server SuiteScript instead of Client SuiteScript for a more robust implementation.

Resetting Default Behavior

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

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

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

- **Use Conditional Defaults on Add:** Similar to the UI, if enabled, related fields are automatically populated with default values when a related value is entered in another field while *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.



Important: You can NOT 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

For 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. Therefore, in a Web services implementation, if the Probability field has already been adjusted based on information unknown to the NetSuite System, it may be undesirable to have the field automatically populated with the conditional default value.

Company	XYZ, Inc	Status	In Discussion
Title		Probability	20.0%
Opportunity #	51	Date Created	1/14/2005
Custom Form	Standard Opportunity	Expected Close	1/14/2005
Sales Rep	A Wolfe	Forecast Type	Omitted
Partner		Currency	
Lead Source		Exchange Rate	
Department		Projected Total	

With conditional defaults enabled, when the Status field is changed, Probability is automatically updated



Note: Endpoints prior to version 2.0.0 also included a **Use Defaults** preference. If enabled, this preference allowed the use of Default values for records. If not enabled, default values were not set, forcing the Web services application to provide all required values. When upgrading to version 2.0.0 or later from an earlier endpoint, you must adjust your code accordingly since this preference is no longer available.

Setting Preferences at the Request Level

In addition to setting preferences to be used for all requests submitted to NetSuite Web services, you can also set preferences at the request level.

To set preferences at the request level, your code must do the following:

- Clear the headers (Java only)
- Create a new SOAPHeaderElement: the same command is used for all preference elements. 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.
- Create the Preference Object: contains the elements you are allowed to set. Elements include:
 - warningAsError
 - useConditionalDefaultsOnAdd
 - useConditionalDefaultsOnUpdate
 - disableMandatoryCustomFieldValidation
 - disableSystemNotesForCustomFields
 - ignoreReadOnlyFields
- Set the header (Java Only)

Sample Code

SOAP Request

```
<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:disableSystemNotesForCustomFields>
</platformMsgs:preferences>
</soap:Header>
```

Java

In this example, a SearchPreference Object is set.

```
NetSuiteBindingStub stub = (NetSuiteBindingStub)aPort;
stub.clearHeaders();
SOAPHeaderElement searchPrefHeader = new
SOAPHeaderElement("urn:messages_2_5.platform.webservices.netsuite.com",
"searchPreferences");
SearchPreferences searchPrefs = new SearchPreferences();
searchPrefs.setPageSize(new Integer(nPageSize));
searchPrefs.setBodyFieldsOnly(isBodyFieldsOnly);
searchPrefHeader.setObjectValue(searchPrefs);
stub.setHeader(searchPrefHeader);
```

C#

In this example, a SearchPreference object is set as well as a warnAsError object.

```
// Set up request level preferences as a SOAP header
Preferences prefs = new Preferences();
_service.preferences = prefs;
// Preference to ask NS to treat all warnings as errors
prefs.warningAsErrorSpecified = true;
prefs.warningAsError = false;
// Invoke search() web services operation
_service.searchPreferences.pageSize = 20;
_service.searchPreferences.pageSizeSpecified = true;
SearchResult response = _service.search( custSearch );
```

Setting the ignoreReadOnlyFields Preference

After getting or initializing a record, it is recommended that you set the ignoreReadOnlyFields preference to *true* when submitting the record using write operations such as add/addList or update/updateList. Setting this preference to *true* reduces the possibility of receiving an INSUFFICIENT_PERMISSION error because a read-only field was mistakenly set and then submitted.

It is also recommended that you set this preference to *true* when using the initialize/initializeList operations. To submit an initialized record without having to remove read-only fields populated during the initialization, set the ignoreReadOnlyFields preference header preference to *true*. When this preference is set to *true*, read-only fields are simply ignored during the Web services request.

Note: For details on initialize operations, see “[initialize / initializeList](#)” on page 146.

Displaying Internal IDs

You can configure NetSuite to display internal ID values on forms in the UI. This is useful during development as a quick reference to verify that the internal ID values submitted on Requests match the records expected as shown in the UI.

To display internal ID values on forms, go to Home > Set Preferences. Click the General tab, and in the Defaults section, click **Show internal IDs**. When this preference is enabled, a given list displays the IDs as one of the columns. For example, List > Relationships > Customers displays the internal ID as the second column.



Note: Changes to these preferences affect the current user only. Also, another way to quickly determine the internal ID value for an item is by hovering over the item in the UI and noting the id value in the URL. For example, when hovering over a specific customer at Lists > Relationships > Customers, you may see something like <https://webservices.netsuite.com/app/common/entity/custjob.nl?id=272> (where 272 is the internal ID).

When working with custom fields, the custom code feature must also be enabled to view Internal IDs for custom fields.

Understanding NetSuite Roles and Permissions

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

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

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



Note: If you are building an integrated application, it is best to create a new role or customize an existing role and grant the minimum set of permissions that are necessary for the client to carry out its functions. It is not recommended that users are granted the Full Access role or that a user should be assigned administrator privileges in your Web services.



Note: If your role has permission to view credit card data on the user interface, you can also retrieve this information through Web Services calls. This is beneficial to integrated applications that use an external credit card processor. Based on your role, you may now be able to retrieve the credit card on file for your customers instead of having to ask them.

Setting a Default Role for a Web Services User

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

- First, any role specified in the Passport object of the request is used. The role defined here must be a valid role contained in the Employee record of the given user. (For information on the Passport object, see “login” on page 79. The Passport object is defined in the [platformCore XSD](#).)
- If a role preference is not set at the request level, then any default Web services role as defined in the Web Services Preference page for the given user is used.
Only one default Web services role can be assigned per user and only roles that contain the Web services permission can be specified as a default Web services role. Note that the user may be assigned a different role than those specified in their Employee record. In other words, a user may have greater or lesser permissions using Web services as compared to the UI.
- If neither the request nor the Web services default role is set, then the user’s default UI role is used, provided it has the Web services permission.
Note: All standard roles have the Web services permission by default when the Web services feature is enabled. Custom roles, however, must be explicitly set to have Web services permissions.

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

1. Click Setup > Integration > Web Services Preferences.
2. Select the desired user from the Name drop-down list.
3. Select the default role to use for Web services requests for this user.
Note: The internal ID for the selected role automatically populates the ID field.
4. Click Add.
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. The Web Services Only role increases the security of an integrated application by prohibiting a UI user from accessing the system with permissions and privileges that are specifically created for a Web services applications.

For example, you may have a Web services application that requires certain employees to have write access to several records. However, you want to prohibit the employees from being able to edit these records directly from within the NetSuite UI. If you assign the Web Services Only role to specified employees, the employees can log in to NetSuite and access the application through Web services, however, the employees cannot switch to their other roles within the system and write/edit/delete these any data-sensitive records.



Important: The Web Services Only role does not appear in the Change Role drop-down 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.



Note: Your account must have the Web services feature enabled for the Web Services Only check box to appear. See “Enabling the Web Services Feature” on page 16 for steps on enabling the Web services feature in NetSuite.

To designate a role as Web Services Only:

1. Click Setup > Users/Roles > Manage Roles.
2. On the Manage Roles list page, select Customize next to the role you want to set as Web Services Only.
3. Select the Web Services Only Role check box.
4. Click Save.

When to Set the Web Services 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. Once the development and testing is complete, the developer can set the Web Services Only role to TRUE for a specified role to prevent users with this role access to the UI with this set of permissions and privileges.



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

Customer Center, Vendor Center, and Partner Center Roles

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



Note: It is not recommended that you customize Customer Center, Partner Center, or Partner Center Roles to have only Web Services Only permissions. For information on the Web Services Only role, see “Setting a Web Services Only Role for a User” on page 28.

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

To remove Web services permissions:

1. Go to Setup > Users/Roles > Manage Roles.
2. Click Customize next to the Customer Center, Vendor Center, or Partner Center role.
3. On the Setup subtab, choose None from the Level drop-down list.

4. Next, click Done.
5. Click Save.

IDs Associated with Roles

The table in this section lists the standard NetSuite roles and the associated internal ID values. You can use these internal ID values in the Passport object, which is used for the login operation. For information on the Passport object and the login operation, see “login” on page 79.

If you have the Show Internal IDs preference on, you can look up the internal ID of a role by going to Setup > Users/Roles > Manage Roles. The role ID appears in the Internal ID column. For instructions on setting the Show Internal IDs preference, see “Displaying Internal IDs” on page 27.



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

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

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

General Concepts

Understanding NetSuite Features

The NetSuite UI allows you to enable or disable certain features. When designing your Web services, it is important to know which features must be enabled in order for the service to execute properly. If the service calls for a function which is NOT available because the associated feature is disabled, a SOAP **InsufficientPermissionFault** occurs causing the entire service to fail.

For example, in the NetSuite UI you can enable or disable the Opportunities feature for CRM. If disabled, a Web service call to add an OpportunityItem will fail.

Web services is itself a feature that **must** be enabled prior to submitting Web services requests. To enable the NetSuite Web Services feature, see [“Enabling the Web Services Feature”](#) on page 16.

PCI Compliance Password Requirements

When using NetSuite’s Credit Card Payments feature, be aware of the Payment Card Industry (PCI) Data Security Standard password requirements. Anyone using the following roles or any custom role with the View Unencrypted Credit Cards permission must change his or her NetSuite password at least every ninety (90) days:

- Administrator
- Accountant
- Bookkeeper
- Controller
- A/R Clerk

If the number of days set in the Password Expiration in Days field on the General Preferences page is less than ninety days, that requirement remains in effect. For example, if your company is set to expire passwords every sixty days, your password expiration date does not change.

However, if your company is set to expire passwords every 120 days, this setting automatically changes to 90 days for employees using these roles.

In addition, passwords for those with access to unencrypted credit card numbers must have a minimum of seven (7) characters. If the number of characters set in the Minimum Password Length field on the General Preferences field is greater, that requirement also remains in effect.

All employees using roles with access to unencrypted credit card numbers will be asked to change passwords to meet the PCI compliance requirements.

Using Web Services Specific Forms

If you are a partner building a generic application, in order to ensure that your applications are account independent, we recommend that you use **custom Web Services specific forms**. Use the custom Web services form for Web services requests by specifying the form in the customForm field. This will alleviate problems associated with customer specific customization to forms used in the UI which may break your Web services application code.



Note: If a form is specified in the WS request then that form is used for validation. If a form is NOT specified then the default preferred form for that record or transaction is used. If a custom form is saved with a record in the UI, that form is NOT used in the Web services request unless it is also the default form for that record type or is explicitly set as the form in the Web services request.

Effects of Account Configuration on Web Services

Form customization in an account or enabling/disabling features may result in required fields being added to various forms used on records and transactions. For each Web service 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.

Working with Sublists

Many NetSuite record types contain sublists of items. For example, transaction records often include the item or expense sublist; entity records often include the addressbook list.

The figures below show examples of these sublists in the UI. The first figure shows the item sublists on a Check record. The second figure shows the addressbook sublist on a Customer record.

Check Find List Go To Register Add To Shortcuts Prev Next

Account: 10000 Bank...t-checking Date: 10.5.2007
 Balance Exchange Rate: 1
 Payee: <Type then tab> Currency: British pound
 Address To Be Printed:
 Check #:
 Amount:
 Memo:

Posting Period: May 2007 Map

Save Auto Fill Reset Memorize Clear Splits Recalc Customize New Body Field

Expenses & Items History

Expenses 0.00 Items 0.00

Add Multiple

Item	Quantity	Units	Description	Serial/Lot Numbers	Expiration Date	Rate	Bin Numbers	Tax Code	Amount	Tax Rate	Options	Location	Customer	Billable
>>														

Add Copy Previous Insert Remove Cancel

Customer Find Customer Accept Payment Generate Statement Prev Next

Create New: Contact Task Phone Call Event Note Email Letter PDF File Subcustomer Subprospect Sublead >>

Customer ID: Adina Fitzpatrick Web Address: <http://thekessmanclinic.med>
 Type: Company Address: Adina Fitzpatrick
 Company Name: The Kessman Clinic 889 Red Roof Road
 Status: CUSTOMER-Closed Won Great Falls, MT Map
 Phone: 504-789-1255 Inactive: No
 Fax: Enable Online Bill Pay: No
 Email: dmueller@thekessmanclinic.med Address:

Edit New Search Print Show Activity View Dashboard

General Info Sales Marketing Support Address Financial Access Custom

Contacts *Activities User Notes *System Notes Messages Files Subcustomers Jobs Time Tracking

New Contact Attach Update Primary Customize View

Contact: Role: View: Default

Edit	Name	Company	Job Title	Phone	Email	Role	Remove

When working with sublists, the method for searching on, updating, or replacing the sublist varies depending on the type of list. There are two types of lists: **Keyed Sublists** and **Non-keyed Sublists**.



Note: When searching records, by default sublists are NOT returned in the result set. This results in optimizing the search. You can change this behavior by setting the search preference for the request. For details, see the description for the search operation in "search" on page 107.

Keyed Sublists

These lists have a **line** field (typically on transaction lists). Individual lines in the list can be searched for by referencing the line value. For example, the OpportunityItemList on the Opportunity record is a keyed sublist.

The value of the **replaceAll** attribute on **Keyed** sublists determines the behavior of your request for these lists.

- **replaceAll = TRUE:** When set to TRUE, the existing list is **replaced** with the line items submitted in your web services request. If you specify a value for the **line** field on a specific line item, then the list is replaced with the list submitted but the line referenced specifically is updated with the additional information.
- **replaceAll = FALSE:** When set to FALSE, line items in your web services request are **added** to the existing list. If you specify a value for the **line** field on a specific line item, then that line is updated in the existing list.

Note: The line field simply references the existing lines numerically, starting at one.

The default value for **replaceAll** is TRUE.

Example

For example, suppose you want 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 field. In the following code, the existing line item is modified to change the quantity.

```
<soapenv:Body>
<platformMsgs:update xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:s0="urn:sales_2_5.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_2_5.platform.webservices.netsuite.com">
<platformMsgs:record xsi:type="s0:Opportunity" internalId="1638">
  <s0:itemList replaceAll="false">
    <s0:item>
      <s0:item internalId="380" type="inventoryItem" />
      <s0:quantity>1.0</s0:quantity>
      <s0:amount>20.0</s0:amount>
    </s0:item>
    <s0:item>
      <s0:line>2</s0:line>
      <s0:quantity>10.0</s0:quantity>
    </s0:item>
  </s0:itemList>
</platformMsgs:record>
</platformMsgs:update>
</soapenv:Body>
```

Non-keyed Sublists

These lists have no line field or internalId element. The line items are recorded in *flat* lists. For example, salesTeamList on the Customer record does not have a key.

Since these sublists are NOT keyed, you can NOT update a specific line in the list. Instead you must interact with the list as a whole. In this way an update operation is similar to the add operation with respect to sublists, and the **replaceALL** is always assumed to be TRUE.

Important: For **Non-keyed** sublists, the **replaceAll** attribute is ignored and behaves as if it were TRUE for all requests.

Example

For example, suppose you want to modify the `addressbookList` associated with a Customer record.

- To update a single address in the list, retrieve the entire list of addresses, change a single address as desired, and then resubmit the entire list.
- To replace the `addressbookList` with a subset of addresses, retrieve the entire list, and then resubmit the subset of addresses. The original `addressbookList` is purged and replaced by the new list containing the subset of addresses.
- To delete the `addressbookList`, submit an empty list.



Note: When adding or updating an `addressbookList` entry, ensure that you do NOT set the `internalId` field as this is read-only.

Example Code

The `addressbookList` item refers to the `ContactAddressbookList` type:

```

<xsd:complexType name="Contact">
  <xsd:complexContent>
    <xsd:extension base="platformCore:Record">
      <xsd:sequence>
        <xsd:element name="entityId" type="xsd:string" minOccurs="0"/>
        ...
        <xsd:element name="addressbookList"
          type="listRel:ContactAddressbookList"
          minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="id" type="xsd:string"/>
    </xsd:extension>
    <!-- primary record key -->
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="ContactAddressbookList">
  <xsd:sequence>
    <xsd:element name="addressbook" type="listRel:ContactAddressbook"
      minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="replaceAll" type="xsd:boolean" default="true"/>
</xsd:complexType>

```

Working with Hidden Fields

The Web services API includes fields that are not visible in the NetSuite UI. These fields are *hidden* in the UI and are used primarily for storing system-generated information such as dates. For example, each time a contact record is created, a hidden `createdDate` field is populated with a system-generated timestamp. It is also possible that certain fields have been hidden through customization.



Important: Hidden fields are not settable in Web services.

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 can NOT be overridden.

Operations such as Add and Update follow the UI **edit** form permissions. Get and Search operations follow the **view** form permission. If a field is not available in **view** mode in the UI, it will not be returned in Web services.

Chapter 3 Platform Features

The SuiteTalk Platform refers to the software infrastructure used to expose NetSuite application functionality via Web services. This section describes some of the core features of the SuiteTalk Platform and includes the following sections:

- **Understanding the WSDL and XSD Structure:** provides an overview of the WSDL and XSD files with information on where each record type is defined.
- **Security:** provides information on the NetSuite security model.
- **Synchronous and Asynchronous Request Processing:** describes how to process requests synchronously versus asynchronously.
- **Using Internal IDs, External IDs, and References:** describes IDs and references and how they are used.
- **Web Services to UI Single Login:** describes how to submit an https POST for single login between Web services applications and the NetSuite UI.
- **Handling Errors:** describes the three different types of exceptions that can be thrown during a web service call — warnings, errors, and faults.

The SuiteTalk Platform has the following characteristics:

- **SOAP encoded Web Services:** the SuiteTalk Platform uses SOAP-based Web services with document style, or Doc style, encoding.
Doc style encoding consists of message-oriented exchanges where an XML schema specified within the SOAP message defines the structure of any messages sent between two applications. RPC exchanges are NOT supported.
- **HTTPS Transport:** currently the only transport supported is HTTPS as defined in the WSDL document.

Understanding the WSDL and XSD Structure

The SuiteTalk Platform has a single WSDL file that describes all of the supported operations and messages. You can view that file here:

https://webservices.netsuite.com/wSDL/v2_6_0/netsuite.wsdl

(where v2_6_0 is the version of the wsdl)

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 tables below show the organization of the XSD files.

Versioning

The SuiteTalk Platform defines versioning for the WSDL, the location of schemas, the namespaces, and end point as follows:

```
WSDL: https://webservices.netsuite.com/wsd/v2_6_0/netsuite.wsdl
<xsd:import namespace="urn:core_2_6.platform.webservices.netsuite.com"
schemaLocation="https://webservices.netsuite.com/xsd/platform/v2_6_0/core.xsd"/>
<port name="NetSuitePort" binding="tns:NetSuiteBinding">
  <soap:address location="https://webservices.netsuite.com/services
NetSuitePort_2_6" />
</port>
```

(where v2_6_0 reflects the current version)

The SuiteTalk Platform maintains backwards compatibility where ever possible. Additions such as new operations or record types result in a minor revision but the namespace will not be incremented. New minor revisions support all prior functions.

However, when backward incompatible changes occur, such as adding a new required field, changing the structure of a data type, or removing an operation or element, the version of the namespace is incremented and your applications must be recompiled to reflect the new version. There is no guarantee that a major new release will support all previous functions.

In the event that an unsupported version is sent in a Web services request, an **InvalidVersionFault** is returned.

NetSuite Messaging XSD Files

These files provide descriptions for the base Web services functions used by all operations.

URL	Alias
https://webservices.netsuite.com/xsd/platform/v2_6_0/core.xsd	platformCore
https://webservices.netsuite.com/xsd/platform/v2_6_0/messages.xsd	platformMsgs
https://webservices.netsuite.com/xsd/platform/v2_6_0/faults.xsd	platformFaults
https://webservices.netsuite.com/xsd/platform/v2_6_0/common.xsd	platformCore

NetSuite Business Records XSD Files

These files provide descriptions for each record type in NetSuite.

URL	Schema Alias	Example Record Types
https://webservices.netsuite.com/xsd/activities/v2_6_0/scheduling.xsd	actSched	Events (CalendarEvent) Tasks Phone Calls
https://webservices.netsuite.com/xsd/general/v2_6_0/communication.xsd	generalCom m	Messages Notes

URL	Schema Alias	Example Record Types
https://webservices.netsuite.com/xsd/transactions/v2_6_0/sales.xsd	tranSales	Opportunities Sales Orders Item Fulfillment Invoice CashSale Estimate
https://webservices.netsuite.com/xsd/transactions/v2_6_0/general.xsd	tranGeneral	Journal Entries
https://webservices.netsuite.com/xsd/lists/v2_6_0/accounting.xsd	listAcct	Other Lists Items
https://webservices.netsuite.com/xsd/lists/v2_6_0/relationships.xsd	listRel	Customers Contacts Employee Group Job Partner Vendor
https://webservices.netsuite.com/xsd/lists/v2_6_0/support.xsd	listSupport	Cases Issues Solution Topic
https://webservices.netsuite.com/xsd/setup/v2_6_0/customization.xsd	setupCustom	Custom Records Custom Fields (All)
https://webservices.netsuite.com/xsd/lists/v2_6_0/employees.xsd	listEmp	Employee
https://webservices.netsuite.com/xsd/documents/v2_6_0/fileCabinet.xsd	docfileCab	File Folder
https://webservices.netsuite.com/xsd/transactions/v2_6_0/bank.xsd	tranBank	Check
https://webservices.netsuite.com/xsd/transactions/v2_6_0/inventory.xsd	tranInvnt	Inventory Adjustment
https://webservices.netsuite.com/xsd/transactions/v2_6_0/purchases.xsd	tranPurch	VendorBill Purchase Order Item Receipt
https://webservices.netsuite.com/xsd/transactions/v2_6_0/customers.xsd	tranCust	CashRefund CustomerPayment ReturnAuthorization CreditMemo
https://webservices.netsuite.com/xsd/lists/v2_6_0/website.xsd	lsitSite	SiteCategory
https://webservices.netsuite.com/xsd/transactions/v2_6_0/employees.xsd	tranEmp	TimeBill

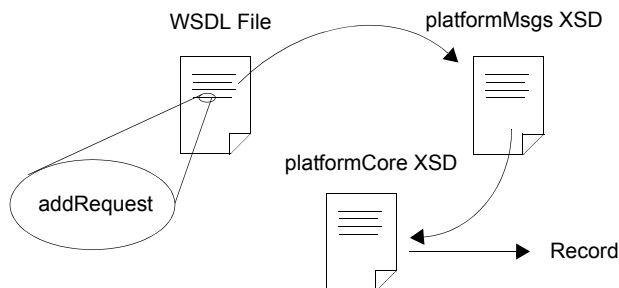
System Constants XSD Files

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

URL	Schema Alias
https://webservices.netsuite.com/xsd/activities/v2_6_0/schedulingTypes.xsd	actSchedTyp
https://webservices.netsuite.com/xsd/general/v2_6_0/communicationTypes.xsd	generalCommTyp
https://webservices.netsuite.com/xsd/lists/v2_6_0/relationshipsTypes.xsd	listRelTyp
https://webservices.netsuite.com/xsd/lists/v2_6_0/supportTypes.xsd	listSupportTyp
https://webservices.netsuite.com/xsd/lists/v2_6_0/accountingTypes.xsd	listAcctTyp
https://webservices.netsuite.com/xsd/transactions/v2_6_0/salesTypes.xsd	tranSalesTyp
https://webservices.netsuite.com/xsd/setup/v2_6_0/customizationTypes.xsd	setupCustomTyp
https://webservices.netsuite.com/xsd/setup/v2_6_0/coreTypes.xsd	platformCoreTyp
https://webservices.netsuite.com/xsd/setup/v2_6_0/faultTypes.xsd	platformFaultsTyp
https://webservices.netsuite.com/xsd/lists/v2_6_0/employeeTypes.xsd	listEmpTyp
https://webservices.netsuite.com/xsd/documents/v2_6_0/fileCabinetTypes.xsd	docFileCabTyp
https://webservices.netsuite.com/xsd/transactions/v2_6_0/inventoryTypes.xsd	invTyp

Example

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



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

```

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

Notice that the element called **platformMsgs:add** is not contained in the WSDL itself but is referenced from the related platformMsgs XSD file. In this case, the platformMsgs alias refers to the xsd file at:

https://webservices.netsuite.com/xsd/platform/v2_6_0/messages.xsd

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

```

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

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

https://webservices.netsuite.com/xsd/platform/v2_6_0/core.xsd

The abstract type Record is defined as:

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



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

Security

NetSuite leverages the latest industry security standards in order to ensure high-levels of security around your business data. All Web service requests are controlled by the same security measures used in the NetSuite UI including:

- Authentication
- Authorization
- Session Management
- Encryption



Important: For information on password requirements associated with the Payment Card Industry (PCI) Data Security Standard, see “[PCI Compliance Password Requirements](#)” on page 31.

Authentication

Authentication is the process of determining the identity of the requester by verifying that they are who they claim to be based upon the credentials they present. The SuiteTalk Platform requires a valid user name, password and account number for authentication. These are provided via the login operation using the Passport object.

```
<complexType name="Passport">
  <sequence>
    <element name="email" type="xsd:string"/>
    <element name="password" type="xsd:string"/>
    <element name="account" type="xsd:string"/>
    <element name="role" type="platformCore:RecordRef" minOccurs="0"/>
  </sequence>
</complexType>
```

After the requester has been successfully authenticated, a new session is created for that user. All Web services operations require authentication.



Note: In order for session information to be successfully transported in SOAP, you must enable support for multiple cookie management in your application. For example, in Microsoft .NET, include the following line:

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

Authorization

Authorization is the process of ensuring that the requester has the appropriate entitlement to perform the requested operation. When a user requests to be authenticated, they also provide a **role** so that they are logged in under that role. For every Web services request, the system uses the role definition to ensure that the user has the required permission for the requested operation as well as the requested record type. The role must be provided in the Passport type via the login operation:

```
<complexType name="Passport">
  <sequence>
    <element name="email" type="xsd:string"/>
    <element name="password" type="xsd:string"/>
    <element name="account" type="xsd:string"/>
    <element name="role" type="platformCore:RecordRef" minOccurs="0"/>
  </sequence>
</complexType>
```

For detailed information on roles and permissions and how the SuiteTalk Platform implements roles and permissions rules refer to [“Understanding NetSuite Roles and Permissions” on page 27](#).

Session Management

After a user has been successfully authenticated, a sessionID is created that must be passed in to each subsequent request. Additional logins are not required as long as the session is active.

Sessions are controlled by the following:

- **Session Timeouts:** Similar to the UI, Web services sessions automatically time-out after 15 minutes of inactivity, requiring a login to resume activity. However, if the server resubmits the cookie during the first 15 minutes, the inactivity timer is reset.
Note: If you need shorter sessions for security reasons, you should create a login route for the client that calls `logout()` when operations are complete.
If you explicitly logout of a session, and then attempt to utilize 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`, is seen.
If you need to eliminate open sessions for security reasons, you should call `logout` upon completion of an operate.
- **Session Limits:** A given login (username/password) is currently limited to two sessions, one through the browser and one through Web services.
For example, if a user attempts to log in to his/her account from two separate browsers, the second browser login terminates the first browser session (assuming the same credentials).
Likewise, if two Web services clients attempt to establish two concurrent sessions, the first session is terminated. However, UI sessions and Web services sessions do not cause termination of each other.
Note that more than two independent concurrent sessions are possible with the purchase of certain products such as Offline Client or Outlook Integration.

Important: Using the same login from multiple client applications, or multiple instances of the same application is NOT supported. However, NetSuite does provide a Web Services Concurrent License for purchase. This license type allows designated employees to use the same credentials five times concurrently before their first session is invalidated. For details see “[Web Services Concurrent License \(Web Services Plus\)](#)” on page 44.

Manually Managing Cookies

When submitting Web services requests, the NetSuite server can accept any SOAP document as long as it is valid against the NetSuite WSDL — it does not matter how the SOAP was generated. When generating a SOAP document from tools or sources that do NOT automatically generate the NetSuite port objects, 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.

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 an InvalidSessionFault — PLEASE_LOGIN_BEFORE_PERFORMING_ACTION.

Example

The following code shows the cookie information returned in the login response.

```
HTTP/1.1 200 OK
Date: Wed, 18 May 2005 18:43:27 GMT
Server: Oracle-Application-Server-10g/9.0.4.0.0 Oracle-HTTP-Server
Set-Cookie: NS_VER=11.0.0; domain=joe.corp.netsuite.com; path=/
Vary: User-Agent
Set-Cookie: JSESSIONID=ac101fae1f4312dfxxx062fc829447eaa00c3dcd70af41d; Path=/
Set-Cookie: lastUser=TSTDRV198400_935_3; Expires=Wed, 25-May-2005 17:42:24 GMT;
Path=/
Cache-Control: private
Keep-Alive: timeout=150, max=1000
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/xml; charset=utf-8
```

For each subsequent request, all of the cookies must be returned in the HTTP header to maintain the session as follows:

Sample Code (C#)

```
NetSuiteService nss = new NetSuiteService();
nss.Url = https://webservices.netsuite.com/services/NetSuitePort_2_0 ;
nss.CookieContainer = new CookieContainer();
```

Sample Code (Java with Axis 1.2 or Axis 1.3)

```
NetSuiteServiceLocator nss = new NetSuiteServiceLocator();
nss.setMaintainSession(true);
```

Web Services Concurrent License (Web Services Plus)

Both new and existing users can be designated as “concurrent Web services users” through the Web services concurrent license, also referred to as the Web Services Plus license. Each Web Services Plus License provides one UI session and up to five concurrent Web services sessions. Any user can be assigned a Web Services Plus License, provided that there are enough licenses available in the account.

The Web Services Plus License does not eliminate the need for session management. Depending on the expected throughput, one or more Web Services Plus Licenses may be needed to meet the required bandwidth. A user who has been assigned the Web Services Plus License will be granted a new JSESSIONID for every login attempt. The sixth login will invalidate the first login for that specific user. This means that users need to track their sessions and still implement pooling and queuing if they believe they will exceed five active sessions at any given time.

The Web Services Concurrent License can only be used by a single instance of an integrated application. For example, this license is not valid for two distinct instances of the same application, one of which uses two concurrent sessions and the second uses three. In this situation, the customer is required to use two Web Services Concurrent Licenses (one per application).



Note: The Web Services Plus License is not available in NetSuite Small Business. Therefore, if you develop an application that relies on this license type, you cannot implement the application in accounts running NetSuite Small Business.



Note: Your account must have the Web services feature enabled before you can assign the Web Services Plus License to a user. See “Enabling the Web Services Feature” on page 16 for steps on enabling Web services.

To assign the Web Services Plus License to a user:

1. Go to Lists > Employees > Employees > New to assign the license to a new employee. If assigning to an existing employee, go to Lists > Employees > Employee and select the employee from the Employees list page.
2. Click the Access tab on the Employee record.
3. Select the Concurrent Web Services User check box.
4. Click Save.

The screenshot shows the 'Employee' form in SuiteTalk. The 'Access' tab is selected, and the 'Concurrent Web Services User' checkbox is checked and circled in red. Other visible fields include 'Give Access', 'Send Notification Email', 'Offline Client Access', 'Password', 'Confirm Password', 'Require Password Change On First Login', 'Inherit IP Rules from Company', and 'IP Address Restriction'. The 'Role' dropdown is set to 'Engineer'.

Encryption

Web services communications are not viewable by a third party as they travel on the Internet. Encryption is implemented using 128-bit encryption with HTTPS/SSL at the transport level. No non-secure Web service requests are granted.

Synchronous and Asynchronous Request Processing

Web services requests can be processed synchronously or asynchronously.

In **synchronous** requests, your client application sends a request to the SuiteTalk Platform where it is processed and a response is returned. The client application handles the response as appropriate.

In **asynchronous** requests, your client application sends a request to the SuiteTalk Platform where it is placed in a processing queue and handled asynchronously with other requests. Note that all available jobs for each polling period will be processed contiguously. There is no enforced waiting period for a job that is available.

Once a job is processed, a job ID is returned in the Web services response. Your client application can then check on the status and result of the request by referencing the job ID.



Note: Asynchronous request JobIDs are valid for 30 days.

Asynchronous equivalents are available for the following operations:

- addList: asyncAddList

- `updateList: asyncUpdateList`
- `deleteList: asyncDeleteList`
- `getList: asyncGetList`
- `search: asyncSearch`
- `initializeList: asyncInitializeList`

When submitting an asynchronous request, the request is similar to the equivalent synchronous call. For example, the following code illustrates the asynchronous request of a list of customer records.

Request

```
<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 instead of the actual record. You can then use the `jobId` to reference the request later.

Response

```
<soapenv:Body>
  <asyncGetListResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
    <asyncStatusResult xmlns="urn:core_2_6.platform.webservices.netsuite.com">
      <jobId>ASYNCEWEBSERVICES_563214_053120061943428686160042948_4bee0685</jobId>
      <status>pending</status>
      <percentCompleted>0.0</percentCompleted>
      <estRemainingDuration>0.0</estRemainingDuration>
    </asyncStatusResult>
  </asyncGetListResponse>
</soapenv:Body>
```

Use the `getAsyncResult` or `checkAsyncStatus` operations to track the asynchronous request.

checkAsyncStatus

The `checkAsyncStatus` operation can be used to check the status of an asynchronous Web services submission. When a `jobId` is submitted, the status, percent complete and estimated remaining duration is returned.

Possible status values that can be returned include:

- `failed`
- `finishedWithErrors`
- `pending`
- `processing`
- `finished`

If the status is failed, finishedWithErrors, or finished, you can use the `getAsyncResult` operation to provide the detailed of the response whether that be the record results, or error and fault messages.



Note: The percent complete and estimated remaining duration, although generally accurate, may vary depending on increases or decreases in database activity during the web services processing.

Request

```
<soap:Body>
<platformMsgs:checkAsyncStatus>
<platformMsgs:jobId>ASYNCWEBSERVICES_563214_053120061943428686160042948_4bee0685
</platformMsgs:jobId>
</platformMsgs:checkAsyncStatus>
</soap:Body>=
```

Response

```
<soapenv:Body>
<checkAsyncStatusResponse
  xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
  <asyncStatusResult xmlns="urn:core_2_0.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>
```

Faults

This operation can throw one of the following faults. See “[Soap Fault Status Codes](#)” on [page 152](#) for more information on faults.

- InvalidSessionFault
- AsyncFault
- UnexpectedErrorFault

getAsyncResult

The `getAsyncResult` operation can be used to retrieve the results of an asynchronous web services submission.

You can use the `getAsyncResult` operation up to 20 times within a 30 day time period to retrieve the results of an asynchronous job. If this limit is exceeded, you will receive an INVALID_JOBID user error for the following reasons:

- You cannot download results of an asynch job more than 20 times within a 30 day period.
- You have attempted to retrieve job results that have already been purged from the system. Async JobIDs are purged from the system every 30 days. If you attempt to download the results 31 days after your initial request, the JobID results will no longer exist.

Faults

This operation can throw one of the following faults. See “Soap Fault Status Codes” on page 152 for more information on faults.

- InvalidSessionFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- AsyncFault
- UnexpectedErrorFault

Monitoring Asynchronous Jobs in the UI

Web services jobs submitted asynchronously can be monitored on the Web Services Job Status page at Setup > Integration > Web Services Process Status.

On this page you can do the following:

- View the status of the job such as percent complete and estimated time.
- View the Job Id. The Job Id, which is also returned in the asynch SOAP response, can be used to programmatically retrieve job status information.
- View the SOAP request or response associated with a job. The response can only be viewed once a job has successfully completed.
- Cancel a Job. A job can only be cancelled if it has not yet successfully completed.



Note: The Job Status page does not automatically refresh. Click refresh in your browser window to get the current status of a job.

Using Internal IDs, External IDs, and References

Each record in NetSuite is uniquely identified by its record type in combination with either a system-generated NetSuite internal ID or an externalID that has been provided at the time of record creation or during an update. Internal and External IDs are NOT unique across different record types. Therefore, both the record type and either the internal ID or the external ID are required to perform an operation on an existing record and both are returned for each record in all operations on records.

Note: Internal IDs are not *reused* in the system. Once an internalID has been assigned, if the associated record is subsequently deleted, the ID is not reused.

The term *reference* or *ref* is used to define a reference to any existing record (record type/id combination) in the system. References are implemented through the RecordRef type described in the following XSD file:

https://webservices.netsuite.com/xsd/platform/v2_6_0/core.xsd

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

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



Important: When referencing records, you must provide the internalID OR the externalID attribute for all update operations but not both in the same operations. If both are provided, the internalID is used to locate the record and the externalID is ignored.

Understanding ExternalIDs

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

ExternalIDs can be set during an Add or Update. ExternalIDs are useful for the following two situations:

- **Maintaining client ID relationships**
In cases where a client application already maintains references between records, set the externalID 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 externalID 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 externalID you can import the customer records in a single API call using the externalID references to the sales reps.



Important: ExternalIDs can be updated through CSV or Web services. Therefore, it is recommended that your organization use a single mechanism for maintaining externalIDs so that externalIDs are not unknowingly updated using two separate methods.

Copying smbXML Handles into ExternalIDs

To facilitate migrations from smbXML to Web services, NetSuite copies over the handles of all new and existing records that were created using smbXML into Web services externalId fields. Existing handles will not be copied if there is already a value in the externalId field.

It is important to note that externalIDs in Web services can be edited. Therefore, it is recommended that once you migrate your data from smbXML to Web services, you should maintain the data using Web services.



Important: Updating an externalID in Web services does not update its corresponding smbXML handle.



Note: You cannot copy the value of an externalID into an smbXML handle.

SuiteTalk 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 externalIDs are provided by the client application, if an externalID is desired for a record, it can be submitted on an Add operation.

Operation	ID Required in Request	ID Returned in Response
login	n/a	n/a
logout	n/a	n/a
add/addList	No	Yes
update/updateList	Yes	Yes
delete/deleteList	Yes	Yes
get/getList	Yes	Yes
getAll	No	Yes
search/searchNext/searchMore	No	Yes

Example Internal ID Usage

The following are partial SOAP messages of operations demonstrating the use of internal IDs.

The add operation

For an add operation, an internal ID is *not* required in the request since the record does not yet exist. The internal ID (in this case 100101) is returned in the response.



Note: There are internal IDs listed in the request, but these internal IDs are *embedded* in the request and do not correspond to the actual record being added (an event) but to other existing records associated with the event record being added.

(Request)

```
<soap:Body>
<platformMsgs:add>
  <platformMsgs:record xsi:type="actSched:CalendarEvent">
    <actSched:title>Web Services Meeting</actSched:title>
    <actSched:organizer internalId="-5" type="calendarEvent"
xsi:type="platformCore:RecordRef">
      <platformCore:name>Mr. Wolfe</platformCore:name>
    </actSched:organizer>
    <actSched:location>Main Conference Room</actSched:location>
    <actSched:attendeeList replaceAll="true"
xsi:type="actSched:CalendarEventAttendeeList">
      <actSched:attendee xsi:type="actSched:CalendarEventAttendee">
        <actSched:sendEmail>>false</actSched:sendEmail>
        <actSched:attendee internalId="21" type="calendarEvent"
xsi:type="platformCore:RecordRef"/>
        <actSched:response>_accepted</actSched:response>
        <actSched:attendance>_optional</actSched:attendance>
      </actSched:attendee>
      <actSched:attendee xsi:type="actSched:CalendarEventAttendee">
        <actSched:sendEmail>>false</actSched:sendEmail>
        <actSched:attendee internalId="27" type="calendarEvent"
xsi:type="platformCore:RecordRef"/>
        <actSched:response>_accepted</actSched:response>
        <actSched:attendance>_optional</actSched:attendance>
      </actSched:attendee>
    </actSched:attendeeList>
  </platformMsgs:record>
</platformMsgs:add>
</soap:Body>
```

```

    </actSched:attendeeList>
  </platformMsgs:record>
</platformMsgs:add>
</soap:Body>

```

(Response)

```

<soapenv:Body>
  <addResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
    <writeResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
      <ns1:status isSuccess="true"
xmlns:ns1="urn:core_2_6.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_2_6.platform.webservices.netsuite.com"/>
    </writeResponse>
  </addResponse>
</soapenv:Body>

```

The 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_2_6.platform.webservices.netsuite.com">
    <writeResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
      <ns1:status isSuccess="true"
xmlns:ns1="urn:core_2_6.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_2_6.platform.webservices.netsuite.com"/>
    </writeResponse>
  </updateResponse>
</soapenv:Body>

```

The deleteList operation

In the following delete request, the internal IDs are required for the request and returned in the response. In this case 3 Event records are deleted (100101, 100102 and 100103).

(Request)

```

<soap:Body>
  <platformMsgs:deleteList>
    <platformMsgs:baseRef internalId="100101" type="calendarEvent"
xsi:type="platformCore:RecordRef"/>
    <platformMsgs:baseRef internalId="100102" type="calendarEvent"
xsi:type="platformCore:RecordRef"/>
    <platformMsgs:baseRef internalId="100103" type="calendarEvent"
xsi:type="platformCore:RecordRef"/>
  </platformMsgs:deleteList>
</soap:Body>

```

(Response)

```

<soapenv:Body>
<deleteListResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
<writeResponseList xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
  <writeResponse>
    <ns1:status isSuccess="true"
xmlns:ns1="urn:core_2_6.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_2_6.platform.webservices.netsuite.com"/>
  </writeResponse>
  <writeResponse>
    <ns3:status isSuccess="true"
xmlns:ns3="urn:core_2_6.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_2_6.platform.webservices.netsuite.com"/>
  </writeResponse>
  <writeResponse>
    <ns5:status isSuccess="true"
xmlns:ns5="urn:core_2_6.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_2_6.platform.webservices.netsuite.com"/>
  </writeResponse>
</writeResponseList>
</deleteListResponse>
</soapenv:Body>

```

The getList operation

In this case, an internal ID is called for each record to be retrieved — in this case, three different Event records. Again, the internal ID is required for the request and returned in the response.

(Request)

```

<soap:Body>
<platformMsgs:getList>
  <platformMsgs:baseRef internalId="100104" type="calendarEvent"
xsi:type="platformCore:RecordRef"/>
  <platformMsgs:baseRef internalId="100105" type="calendarEvent"
xsi:type="platformCore:RecordRef"/>
  <platformMsgs:baseRef internalId="100106" type="calendarEvent"
xsi:type="platformCore:RecordRef"/>
</platformMsgs:getList>
</soap:Body>

```

(Response)

```

<soapenv:Body>
<getListResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
<readResponseList xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
<readResponse>
  <ns1:status isSuccess="true"
xmlns:ns1="urn:core_2_6.platform.webservices.netsuite.com"/>
  <record internalId="100104" xsi:type="ns2:CalendarEvent"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns2="urn:scheduling_2_6.activities.webservices.netsuite.com">
    <ns2:title>Customization Meeting</ns2:title>
    <ns2:organizer internalId="-5">
      .....[more fields]
  </record>
</readResponse>

```

```

<readResponse>
  <ns8:status isSuccess="true"
xmlns:ns8="urn:core_2_6.platform.webservices.netsuite.com"/>
  <record internalId="100105" xsi:type="ns9:CalendarEvent"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns9="urn:scheduling_2_6.activities.webservices.netsuite.com">
    <ns9:title>Web Services Meeting (Records)</ns9:title>
    <ns9:organizer internalId="-5">
      .....[more fields]
    </record>
</readResponse>
<readResponse>
  <ns15:status isSuccess="true"
xmlns:ns15="urn:core_2_6.platform.webservices.netsuite.com"/>
  <record internalId="100106" xsi:type="ns16:CalendarEvent"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns16="urn:scheduling_2_6.activities.webservices.netsuite.com">
    <ns16:title>Web Services Meeting</ns16:title>
    <ns16:organizer internalId="-5">
      .....[more fields]
    </record>
</readResponse>
</readResponseList>
</getListResponse>
</soapenv:Body>

```

The search operation

For the search operation, an internal ID is not required for the request but is returned for each record found that matches the specified criteria. In this case, two Event records are returned for a search request calling for each Event record that contains Web Services in the title field.

(Request)

```

<soap:Body>
<platformMsgs:search>
  <platformMsgs:searchRecord xsi:type="actSched:CalendarEventSearch">
    <actSched:title operator="contains"
xsi:type="platformCore:SearchStringField">
      <platformCore:searchValue>Web Services</platformCore:searchValue>
    </actSched:title>
  </platformMsgs:searchRecord>
</platformMsgs:search>
</soap:Body>

```

(Response)

```

<soapenv:Body>
<searchResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
<searchResult xmlns="urn:core_2_6.platform.webservices.netsuite.com">
  <status isSuccess="true"/>
  <totalRecords>2</totalRecords>
  <pageSize>10</pageSize>
  <totalPages>1</totalPages>
  <pageIndex>1</pageIndex>
  <recordList>
    <record internalId="100105" xsi:type="ns1:CalendarEvent"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns1="urn:scheduling_2_6.activities.webservices.netsuite.com">
      <ns1:title>Web Services Meeting (Records)</ns1:title>
      <ns1:organizer internalId="-5">
        .....[more fields]
      </record>
    <record internalId="100106" xsi:type="ns2:CalendarEvent"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```

```

xmlns:ns2="urn:scheduling_2_6.activities.webservices.netsuite.com">
  <ns2:title>Web Services Meeting</ns2:title>
  <ns2:organizer internalId="-5">
    ....[more fields]
  </record>
</recordList>
</searchResult>
</searchResponse>
</soapenv:Body>

```

Web Services to UI Single Login

With SuiteFlex Web services, a single login can be provided and access to both the Web services application and the NetSuite UI maintained. The Web services application submits a POST to a specific wslogin page URL with credentials and a taskId identifying the desired NetSuite page. The wslogin page then authenticates the request, provides a new session and redirects to the requested page.



Important: Using the POST mechanism ensures security because the user's credentials will be encrypted using https and, therefore, will NOT be vulnerable during transmission.

The following table lists the POST parameters that can be submitted.

Parameter	Type	Required for Authentication	Description
email	text	Yes	Must reflect a valid email address of an entity in your account.
password	password	Yes	The password of the entity associated with the email provided.
taskId	text	Yes	For a list of currently exposed NetSuite TaskIDs, refer to "Task IDs" on page 175. To find the TaskID for a given page while in NetSuite, view the source of the page and search for main_help_anchor . A snippet similar to the following is found: 'main_help_anchor' href="#" onclick="nlPopupHelp('LIST_SAVEDSEARCH','Full'); (where LIST_SAVEDSEARCH is the taskID of the page)
id	text	Yes	The ID of a record type available in your NetSuite account.
role	text	Optional	Sets the role for the login. This must be a valid role for the user logging in.
e	text	Optional	If set to T, the record is displayed in edit mode. Any other value is ignored and causes the record to be displayed in view mode.

The post URL is: <https://system.netsuite.com/app/webservices/wslogin.nl?c=#####>
(where ##### is your account number).



Note: Since every NetSuite user is restricted to a single UI session at any given time, initiating a new browser session using this mechanism will invalidate an existing session.

Sample HTML

The following HTML provides a simple login page where clicking Submit sends the user to the Item page in edit mode with administrator privileges. Edit the email and password input fields with values from your account to successfully run this sample.

```
<html>
<body>
<form action="https://system.netsuite.com/app/webservices/wslogin.nl?c#####"
method="POST">
    email <input type="text" name="email"/>
    <br/>password <input type="password" name="password"/>
    <br/>role <input type="text" size="50" name="role"/>
    <br/>task id <input type="text" size="50" value="EDIT_ITEM"
name="taskid"/>
    <br/>record id <input type="text" size="3" value="123" name="id"/>
    <br/>edit flag <input type="text" size="1" value="T" name="e"/>
    <br/><input type="submit"/>
</form>
</body>
</html>
```

Handling Errors

The SuiteTalk Platform supports three different exception types that are returned in the SOAP response in the event of a problem. The client can then take the appropriate action based on the error code or fault that is received.

- **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 “[Setting Company-Wide Preferences](#)” on page 22.
- **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 in order to prevent a subsequent error or to ensure better data quality. In the Netsuite UI, a warning is presented to the user through a dialog box and generally requires an action from the user. Since there is no interaction of this nature in the Web services model, the request must specify what to do in the case of a warning. The options are:

- Ignore the warning and submit the record to the database
- Heed the warning and abort the submission — treat as an error

You can set a company wide preference on the Web Services Preferences page on how to handle warnings or you can specify how warnings should be handled in a specific request. Request level preferences override company-wide preferences. Refer to [“Setting Your Web Services Preferences” on page 22](#) 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):

```
<addResponse xmlns="urn:messages.platform_2_6.webservices...">
  <writeResponse xmlns="urn:messages.platform_2_6...">
    <ns1:status isSuccess="true" xmlns:ns1="urn:core.platform_2_6...">
      <ns1:statusDetail type="WARN">
        <ns1:code>WARNING</ns1:code>
        <ns1:message>Without a zip/postal code it will not be possible to use
          this address with 3rd Party shippers. Click Cancel to edit the address.
        </ns1:message>
      </ns1:statusDetail>
    </ns1:status>
    <ns2:recordRef internalId="50" type="customer"
      xmlns:ns2="urn:core.platform_2_6..." />
  </writeResponse>
</addResponse>
```

The above customer was added because no error occurred. However, future errors could be avoided by responding appropriately to the warning message.

For a detailed list of all warning messages and the associated codes generated by NetSuite, see [“Warning Status Codes” on page 174](#).

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.

```
<updateListResponse xmlns="urn:messages.platform_2_6...">
  <listWriteResponse>
    <writeResponse>
      <ns1:status isSuccess="true" xmlns:ns1="urn:core.platform_2_6..." />
      <ns2:recordRef internalId="100010" type="event"
        xmlns:ns2="urn:core.platform..." />
    </writeResponse>
    <writeResponse>
      <ns3:status isSuccess="false" xmlns:ns3="urn:core.platform_2_6...">
        <ns3:statusDetail type="ERROR">
          <ns3:code>USER_EXCEPTION</ns3:code>
          <ns3:message>Invalid record:
            type=event, id=100015, scompid=TSTDRV96
          </ns3:message>
        </ns3:statusDetail>
      </ns3:status>
      <ns4:recordRef internalId="100015" type="event"
        xmlns:ns4="urn:core.platform_2_6..." />
    </writeResponse>
```

```

    </listWriteResponse>
  </updateListResponse>

```

For a detailed list of all error messages and the associated codes, see [“Error Status Codes” on page 153](#).

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.

```

<soapenv:Fault>
  <faultcode>soapenv:Server.userException</faultcode>
  <faultstring>com.netledger.dto.faults.InvalidSessionFault: Your connection has
    timed out. Please log in again.
</faultstring>
  <detail>
    <invalidSessionFault
      xmlns="urn:faults.platform_2_6.webservices.netsuite.com">
      <code>INVALID_SESSION</code>
      <message>Your connection has timed out. Please log in again.</message>
    </invalidSessionFault>
    <ns1:stackTrace xmlns:ns1="http://xml.apache.org/
      axis">com.netledger.dto.faults.InvalidSessionFault: Your connection has timed
      out. Please log in again.
    </ns1:stackTrace>
  </detail>
</soapenv:Fault>

```

SOAP uses the detail element to capture the error code through the code element and the error message through the message element. The `faultcode` and `faultstring` are automatically populated by the server.

Following is an example of a SOAP fault named `InvalidCredentialsFault` for an invalid e-mail address that is returned as part of the login operation.

```

<soapenv:Fault>
  <faultcode>soapenv:Server.userExveption</faultcode>
  <faultstring>com.netledger.dto.faults.InvalidCredentialsFault: You have
    entered an
    invalid e-mail address or account number. Please try again.</faultstring>
  <detail>
    <InvalidCredentialsFault
      xmlns="urn:faults.platform_2_6.webservices.netsuite.com">
      <code>INVALID_USERNAME</code>
      <message>You have entered an invalid e-mail address or account number.
        Please try again.</message>
    </InvalidCredentialsFault>
  </detail>
</soapenv:Fault>

```

For a detailed list of all fault messages and the associated codes, see [“Soap Fault Status Codes” on page 152](#).

Chapter 4 Types

This section describes the following:

- **Built-in Types:** primitive or derived datatypes defined in the XML Schema specification
- **Complex Types:** NetSuite derived datatypes
- **Custom Field Types:** datatypes used for custom fields
- **Search Types:** datatypes used for search records
- **Platform Enumerations:** datatypes used to populate system defined lists

A WSDL document in Web Services uses the XML Schema in order to define the structure, semantics and constraints of the messages sent between the sender and recipient. The XML Schema is a W3C standard.

There is a strong analogy between object-oriented programming and XML Schema. A type defined in XML Schema can be used to represent an instance of that type in an XML document. Therefore, the body of a SOAP message consists of a combination of such instances of an XML Schema type.

Built-in Types

Built-in (or primitive) types are those that are not defined in terms of other datatypes. They are used as a standardized way to define, send, receive and interpret basic data types in SOAP messages. Primitive data types used in the SuiteTalk Platform can be modified for display purposes. For example, although a price field may be passed in the SOAP messages using an integer primitive data type, the NetSuite UI may format the value with a currency symbol for display purposes.

Of the extensive set of built-in (or primitive) types provided by the XML Schema language specification, the SuiteTalk Platform implementation uses the following built-in types. For detailed information on XML Schema built-in types, refer to <http://www.w3.org/TR/xmlschema-2/>.

- **string:** represents character strings in XML
- **int:** derived from the decimal type and represents the standard concept of the integer numbers
- **double:** is patterned after the IEEE double-precision 64-bit floating point type
- **boolean:** represents the concept of binary-valued logic: {true, false}. In NetSuite, a boolean field can either be set to true or false. If your Web service does not explicitly set the field as true or false the field returns false.
- **dateTime:** represents integer-valued year, month, day, hour and minute properties, a decimal-valued second property and a boolean timezoned 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 the timezone offset.

Note: SOAP encoding is NOT sensitive to your NetSuite Timezone preference as defined in the user preferences. When using Axis, an Axis Web Services client encodes in GMT, regardless of how the machine/JVM is configured. Netsuite will generally encode in PST.

- **date:** represents the top-open intervals of exactly one day in length on the timelines of `dateTime`, beginning on the beginning moment of each day
- **time:** represents an instant of time that recurs every day



Note: Date, time and `dateTime` types listed above conform to the ISO 8601 standard for international dates.

Please see <http://www.w3.org/TR/2001/RED-xmlschema-2-20010502/datatypes#dateTime> for more details.

Complex Types

Complex types are structured types built by aggregating elements of simple or previously defined complex types. NetSuite complex types are defined in the platform core and platform core types XSDs. All NetSuite record types ultimately reference one of these complex types.

NetSuite complex types include the following:

- Passport
- Record
- RecordList
- BaseRef
- RecordRef
- CustomRecordRef
- ListOrRecordRef
- Status
- StatusDetail
- NullField
- ReadResponse
- ListReadResponse
- ListWriteResponse
- WriteResponse

Passport

The Passport type is used by the login operation to encapsulate the security credentials required for authentication. The passport type is defined in the `core.xsd`.

Field Name	XML Schema Type	Req	Notes
email	xsd:string	Y	
password	xsd:string	Y	
account	xsd:string	Y	Must correspond to a valid NetSuite account number
role	RecordRef	Y	A role is a set of permissions that lets a user access specific areas of data. If not set, NetSuite uses the default Web Services role set for this user.

The default Web services role for a given user is dependent on the values set when setting the permissions for given role. These are set through the UI in Setup > Users/Roles > Manage Roles.

Record

The Record type is an abstract type used as the parameter for the add, addList, delete, deleteList, update and updateList operations. It is also returned in the get, getList, search, searchMore and searchNext operations. All business object types extend the Record type. The record type is defined in core.xsd.

Field Name	XML Schema Type	Req	Notes
id	xsd:string (attribute)	N	
xsi:type	xsi:type (attribute)	N	This is a field (attribute) that is automatically implemented by the XML Schema. The value should represent the concrete Record type such as Customer or Event.
nullFieldList	NullFields[]	N	A list of fields that are to be set to null explicitly in the update operation.

RecordList

The RecordList type is an array of the Record type. The recordList type is defined in core.xsd.

Field Name	XML Schema Type	Req	Notes
record	Record []	N	

BaseRef

The BaseRef type is an abstract type used to reference any existing record in NetSuite including other business records and custom records. The BaseRef type is defined in core.xsd.

Field Name	XML Schema Type	Req	Notes
name	BaseRef	N	

RecordRef

The RecordRef type is used to reference an existing record in NetSuite in get operations. Typically, a RecordRef references one of the following:

- Another business object such as a customer or sales order
- An entry in a user defined list such as the category list for contacts or sales tax items

The recordRef type descends from BaseRef and is defined in core.xsd.

Field Name	XML Schema Type	Req	Notes
internalId	xsd:string (attribute)	Y	See "Using Internal IDs, External IDs, and References" on page 48.
type	xsd:string (attribute)	N	Reference to a value in a Web services only system list. If the type is known by context, the type is NOT required.
name	xsd:string	N	This is a read-only field that is populated by NetSuite when it's a part of a get or search response. If this field is populated during a write operation, it will be ignored.

CustomRecordRef

The CustomRecordRef type is used to reference any existing custom record in NetSuite. The CustomRecordRef type descends from BaseRef and is defined in core.xsd.



Important: Setting the RecordRef.type to **customRecord** on an add does NOT return a CustomRecord. You must use CustomRecordRef. The CustomRecordRef has a typeId to indicate which kind of CustomRecord it is.

Field Name	XML Schema Type	Req	Notes
internalId	xsd:string (attribute)	Y	References the primary record internal Id. This Id corresponds to the type of custom record. For a list of possible values, see "Record Internal IDs" on page 12 in the Web Services Records Guide.
typeId	xsd:string	Y	References the custom record type Id.
type	xsd:string (attribute)	N	Reference to a value in a Web services only system list.
name	xsd:string	N	

ListOrRecordRef

The listOrRecordRef type is defined in core.xsd.

Field Name	XML Schema Type	Req	Notes
internalId	xsd:string (attribute)	Y	See “Using Internal IDs, External IDs, and References” on page 48.
externalId	xsd:string (attribute)	N	Use to reference records by their external ID in select and multi-select custom fields.
typeId	xsd:string (attribute)	N	Reference to a value in a Web services only system list.
name	xsd:string	N	This is a read-only field that is populated by NetSuite when it’s a part of a get or search response. If this field is populated during a write operation, it is ignored.

Status

The Status type contains an array of objects of type StatusDetail. The status type is defined in core.xsd.

Field Name	XML Schema Type	Req	Notes
statusDetail	StatusDetail []	N	Used to capture the specific details for the status. See StatusDetail .
isSuccess	xsd:Boolean (attribute)	Y	Indicates whether the status is successful or not. If false, one or more statusDetail objects are populated.

StatusDetail

The StatusDetail type is used to capture the specific details for the status. The statusDetail type is defined in core.xsd.

Field Name	XML Schema Type	Req	Notes
code	xsd:string	Y	The status code. See “Handling Errors” on page 150 for a listing of codes.
message	xsd:string	Y	The detailed message for this status. See “Handling Errors” on page 150 for details.
type	xsd:string		Reference to a value in a Web services only system list. Values: error, warning See “Handling Errors” on page 150 for details.

NullField

The NullField type is defined in core.xsd. It contains the following fields.

Field Name	XML Schema Type	Req	Notes
name	xsd:string	Y	Name of the field to be null. The specified name must exactly match an existing field name.

ReadResponse

The ReadResponse type is used by the following read operations.

- The getResponse output message for the get operation.
- The searchResponse output message for the search operations.

These types have a field named readResponse of the type ReadResponse. The ReadResponse type is defined in message.xsd.

Field Name	XML Schema Type	Req	Notes
status	Status	Y	
recordRef	RecordRef	N	

ListReadResponse

The ListReadResponse type is used by the following operations.

- The GetListResponse output message for the getList operation.

These types have a field named response of type ListReadResponse. The ListReadResponse type is defined in message.xsd.

Field Name	XML Schema Type	Req	Notes
response	ReadResponse[]	Y	An array of ReadResponse types.

ListWriteResponse

The ListWriteResponse type is used by the following operations.

- The AddListResponse output message for the addList operation.
- The UpdateListResponse output message for the updateList operation.
- The DeleteListResponse output message for the deleteList operation.

These types have a field named response of type ListWriteResponse. The ListWriteResponse type is defined in message.xsd.

Field Name	XML Schema Type	Req	Notes
response	WriteResponse[]	Y	An array of WriteResponse types.

WriteResponse

The WriteResponse type is used by the following operations:

- The AddResponse output message for the add operation
- The UpdateResponse output message for the update operation
- The DeleteResponse output message for the delete operation

These types have a field named `writeResponse` of type `WriteResponse`. The `WriteResponse` type is defined in `message.xsd`.

Field Name	XML Schema Type	Req	Notes
status	Status	Y	
recordRef	RecordRef	N	

Custom Field Types

Custom fields are represented by the type `CustomFieldRef` which is an abstract type. The table below contains a list of concrete custom field types that extend the `CustomFieldRef` type. Each type is followed by its corresponding type in the UI.

XML Schema Type	Custom Field Type in UI
<code>IntCustomFieldRef</code>	Integer Number
<code>DoubleCustomFieldRef</code>	Decimal Number
<code>StringCustomFieldRef</code>	Free-Form Text Text Area Phone Number E-mail Address Hyperlink Rich Text
<code>BooleanCustomFieldRef</code>	Check Box
<code>DateCustomFieldRef</code>	Date
<code>SelectCustomFieldRef</code>	List/Record
<code>MultiSelectCustomFieldRef</code>	Multiple Select

CustomFieldRef

The `CustomFieldRef` type is an abstract type.

Field Name	XML Schema Type	Req	Notes
internalId	xsd:string (attribute)	Y	References a unique instance of a custom field type.

To locate the internal ID for a given custom field in the UI, go to Setup > Customization > *[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.

IntCustomFieldRef

The `IntCustomFieldRef` type extends the `CustomFieldRef` abstract type.

Field Name	XML Schema Type	Req	Notes
value	xsd:int	Y	

DoubleCustomFieldRef

The DoubleCustomFieldRef type extends the CustomFieldRef abstract type.

Field Name	XML Schema Type	Req	Notes
value	xsd:double	Y	

BooleanCustomFieldRef

The BooleanCustomFieldRef type extends the CustomFieldRef abstract type.

Field Name	XML Schema Type	Req	Notes
value	xsd:boolean	Y	

StringCustomFieldRef

The StringCustomFieldRef type extends the CustomFieldRef abstract type.

Field Name	XML Schema Type	Req	Notes
value	xsd:string	Y	

DateCustomFieldRef

The DateCustomFieldRef type extends the CustomFieldRef abstract type.

Field Name	XML Schema Type	Req	Notes
value	xsd:datetime	Y	

SelectCustomFieldRef

The SelectCustomFieldRef type extends the CustomFieldRef abstract type. This references a single ListOrRecordRef and also requires an InternalId attribute to indicate the field name.

Field Name	XML Schema Type	Req	Notes
value	ListorRecordRef	Y	A single ListOrRecordRef.

MultiSelectCustomFieldRef

The MultiSelectCustomFieldRef type extends the CustomFieldRef abstract type. This references an array of ListOrRecordRefs and also requires an InternalId attribute to indicate the field name.

Field Name	XML Schema Type	Req	Notes
value	ListOrRecordRef[]	Y	An array of type RecordRef

CustomFieldList

Field Name	XML Schema Type	Req	Notes
value	customFieldRef []	Y	An array of type customFieldRef. The actual entries in the array will be of a concrete type that extends customFieldRef.

The following is an XML excerpt from a SOAP body that illustrates a custom field list that contains all the available custom field types.

```
<customFieldList>
  <customField xsi:type="BooleanCustomFieldRef" internalId="CUSTEVENT7">
    <value>true</value>
  </customField>
  <customField xsi:type="DateCustomFieldRef" internalId="CUSTEVENT4">
    <value>2003-01-20T18:47:00</value>
  </customField>
  <customField xsi:type="DoubleCustomFieldRef" internalId="CUSTEVENT2">
    <value>23.465</value>
  </customField>
  <customField xsi:type="IntCustomFieldRef" internalId="CUSTEVENT5">
    <value>23</value>
  </customField>
  <customField xsi:type="ListCustomFieldRef" internalId="CUSTEVENT3">
    <value internalId="offsite"/>
  </customField>
  <customField xsi:type="StringCustomFieldRef" internalId="CUSTEVENT12">
    <value>John Williams</value>
  </customField>
  <customField xsi:type="MultiSelectCustomFieldRef" internalId="CUSTEVENT11">
    <value internalId="important"/>
    <value internalId="strategic"/>
  </customField>
</customFieldList>
```

Search Types

Searching within Web services is implemented such that every implemented record type has a corresponding search record type. For example, there is a CustomerSearch type in addition to a Customer record type.

Search XML Schema Types

The following sections define the available search types. Every search field within a search object type must belong to one of these search types.

SearchPreferences

Field Name	XML Schema Type	Req	Notes
bodyFields Only	boolean	Y	Defaults to True
pageSize	int		

SearchRequest

Field Name	XML Schema Type	Req	Notes
preferences	boolean	Y	
searchRecord			

SearchResult

Field Name	XML Schema Type	Req	Notes
totalRecords	int	N	
pageSize	int	N	
totalPages	int	N	
pageIndex	int	N	

SearchStringField

Field Name	XML Schema Type	Req	Notes
operator	platformCoreTyp: SearchStringFieldOperator (attribute)	Y	Reference to a value in a system list. For more information on available values, see "Platform Enumerations" on page 74.
searchValue	xsd:string	Y	

SearchBooleanField

Field Name	XML Schema Type	Req	Notes
operator	xsd:boolean	Y	The available values are true or false.

SearchDoubleField

Field Name	XML Schema Type	Req	Notes
operator	platformCoreTyp: SearchDoubleFieldOperator (attribute)	Y	Reference to a value in a system list. For more information on available values, see "Platform Enumerations" on page 74.

Field Name	XML Schema Type	Req	Notes
searchValue	xsd:double	Y	
searchValue2	xsd:double	N	If the operator is between or notBetween searchValue2 must be populated.

SearchIntField

Field Name	XML Schema Type	Req	Notes
operator	platformCoreTyp: SearchIntFieldOperator (attribute)	Y	Reference to a value in a system list. For more information on available values, see “Platform Enumerations” on page 74.
searchValue	xsd:int	Y	
searchValue2	xsd:int	N	If the operator is between or notBetween searchValue2 must be populated.

SearchDateField

Field Name	XML Schema Type	Req	Notes
operator	platformCoreTyp: SearchDate (attribute)	Y	Reference to a value in a system list. For more information on available values, see “Platform Enumerations” on page 74.
predefinedSearchValue	platformCoreTyp: SearchDate	N	Reference to a value in a system list. For more information on available values, see “Platform Enumerations” on page 74.
searchValue	xsd:date	N	Either predefinedSearchValue or searchValue should be populated.
searchValue2	xsd: date	N	If the operator is between or notBetween searchValue2 must be populated.

SearchMultiSelectField

This search type is used to specify a list of one or more internal IDs that reference other user defined records in the system.

Field Name	XML Schema Type	Req	Notes
operator	platformCoreTyp: SearchMultiSelectFieldOperator (attribute)	Y	Reference to a value in a system list. For more information on available values, see "Platform Enumerations" on page 74.
searchValue	platformCore: RecordRef	Y	An array of type SearchMultiSelectRefValue.

SearchEnumMultiSelectField

This search type is used to specify a list of one or more system defined constants.

Field Name	XML Schema Type	Req	Notes
operator	platformCoreTyp: SearchEnumMultiSelectField Operator (attribute)	Y	Reference to a value in a system list. For more information on available values, see "Platform Enumerations" on page 74.
searchValue	xsd:string	Y	

Search Custom Field XML Schema Types

The following sections define the available search types for custom fields. Every search field within a search object type must belong to one of these search types.

SearchCustomField

This is an abstract type.

Field Name	XML Schema Type	Req	Notes
id	xsd:string (attribute)	Y	References a unique instance of a custom field

SearchStringCustomField

The SearchStringCustomField type extends the SearchCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
id	xsd:string (attribute)	Y	Inherited from the SearchCustomField. Reference a unique instance of a custom field.
operator	platformCoreTyp: SearchStringFieldOperator (attribute)	Y	The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see "Platform Enumerations" on page 74.
searchValue	xsd:string	Y	

SearchBooleanCustomField

The SearchBooleanCustomField type extends the SearchCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
id	xsd:string (attribute)	Y	Inherited from the SearchCustomField. Reference a unique instance of a custom field.
operator	platformCoreTyp: SearchBooleanFieldOperator (attribute)	Y	The type is an enumeration type that restricts the value to true or false.

SearchIntCustomField

The SearchIntCustomField type extends the SearchCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
id	xsd:string (attribute)	Y	Inherited from the SearchCustomField. Reference a unique instance of a custom field.
operator	platformCoreTyp: SearchIntFieldOperator (attribute)	Y	The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see “Platform Enumerations” on page 74 .
searchValue	xsd:int	Y	
searchValue2	xsd:int	N	

SearchDoubleCustomField

The SearchDoubleCustomField type extends the SearchCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
id	xsd:string (attribute)	Y	Inherited from the SearchCustomField. Reference a unique instance of a custom field.
operator	platformCoreTyp: SearchDoubleFieldOperator (attribute)	Y	The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see “Platform Enumerations” on page 74 .
searchValue	xsd:double	Y	
searchValue2	xsd:double	N	

SearchDateCustomField

The SearchDateCustomField type extends the SearchCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
id	xsd:string (attribute)	Y	Inherited from the SearchCustomField. Reference a unique instance of a custom field.
operator	platformCoreTyp: SearchDateFieldOperator (attribute)	Y	The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see "Platform Enumerations" on page 74.
searchValue	xsd:date	Y	
searchValue2	xsd:date	Y	

SearchSelectCustomField

The SearchSelectCustomField type extends the SearchCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
id	xsd:string (attribute)	Y	Inherited from the SearchCustomField. Reference a unique instance of a custom field.
operator	platformCoreTyp: SearchSelectFieldOperator (attribute)	Y	The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see "Platform Enumerations" on page 74.
searchValue	xsd:string	Y	

SearchMultiSelectCustomField

The SearchMultiSelectCustomField type extends the SearchCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
id	xsd:string (attribute)	Y	Inherited from the SearchCustomField. Reference a unique instance of a custom field.
operator	platformCoreTyp: SearchMultiSelectFieldOperator (attribute)	Y	The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see "Platform Enumerations" on page 74.
searchValue	RecordRef	Y	

SearchEnumMultiSelectCustomField

The SearchEnumMultiSelectCustomField type extends the SearchCustomField abstract type.

Field Name	XML Schema Type	Req	Notes
id	xsd:string (attribute)	Y	Inherited from the SearchCustomField. Reference a unique instance of a custom field.
operator	platformCoreType: SearchEnumMultiSelectFieldOperator (attribute)	Y	The type is an enumeration type that restricts the value to a predefined list. For more information on available values, see "Platform Enumerations" on page 74.
searchValue	RecordRef	Y	

SearchCustomFieldList

Field Name	XML Schema Type	Req	Notes
customField	platformCore: SearchCustomField []	Y	

Sample Code

SOAP Request — Opportunity Search

Following is an example that contains an excerpt of the SOAP body that illustrates an opportunity search containing several search field types.

```
<opportunitySearch>
  <projectedTotal operator="lessThan">
    <searchValue>100000</searchValue>
  </projectedTotal>
  <title operator="contains">
    <searchValue>Enterprise</searchValue>
  </title>
  <createdDateRange operator="between">
    <fromValue>2003-10-02</fromValue>
    <toValue>2003-10-12</toValue>
  </createdDateRange>
  <opportunityStatusList operator="anyOf">
    <searchValue>inProgress</searchValue>
    <searchValue>closedWon</searchValue>
  </opportunityStatusList>
  <customFieldList>
    <customField xsi:type="SearchSelectCustomField" internalId="SalesEngineer"
      operator="equals">
      <searchValue>Buddy Williams</searchValue>
    </customField>
    <customField xsi:type="SearchBooleanCustomField"
      internalId="hasSalesEngineer"
      operator="true"/>
    <customField xsi:type="SearchStringCustomField" internalId="DemoNotes"
      operator="startsWith">
      <searchValue>CRM</searchValue>
    </customField>
    <customField xsi:type="SearchMultiSelectCustomField"
      internalId="ProductAreas">
```

```

operator="noneOf">
  <searchValue>Inventory</searchValue>
  <searchValue>Warehousing</searchValue>
</customField><
</customFieldList>
</opportunitySearch>

```

Platform Enumerations

The following record and search types are used throughout Web services to populate system defined lists. The tables below outline the available values that should be used to populate these fields in your Web services requests. These enumerations are defined in the platformCoreTyp XSD.

Enumerations	Record Types	
	RecordType	GetAllRecordTypes
calendarEvent	X	
task	X	
customer	X	
contact	X	
message	X	
note	X	
supportCase	X	
opportunity	X	
customRecord	X	
customerCategory	X	X
contactCategory	X	X
priceLevel	X	X
winLossReason	X	X
term	X	X
noteType	X	X
paymentMethod	X	X
leadSource	X	X

Enumerations	SearchString FieldOperator	SearchIntField Operator	SearchDouble FieldOperator	SearchPeriod Field	SearchTime FieldOperator
is	X				
isNot	X				
startsWith	X				
doesNotStartWith	X				
contains	X				

Enumerations	SearchString FieldOperator	SearchIntField Operator	SearchDouble FieldOperator	SearchPeriod Field	SearchTime FieldOperator
doesNotContain	X				
equalTo		X	X	X	
lessThan		X	X		
greaterThan		X	X		
lessThanOrEqualTo		X	X		
greaterThanOrEqualTo		X	X		
notEqualTo		X	X		
notLessThan		X	X		
notGreaterThan		X	X		
notLessThanOrEqualTo		X	X		
notGreaterThanOrEqualTo		X	X		
between		X	X		X
notBetween		X	X		
empty		X	X		
notEmpty		X	X		

Enumerations	SearchDate FieldOperator	SearchEnum MultiSelect FieldOperator	SearchMulti SelectField Operator	SearchDate
anyOf		X	X	
noneOf		X	X	
on	X			
before	X			
after	X			
onOrBefore	X			
onOrAfter	X			
within	X			
empty	X			
notOn	X			
notBefore	X			
notAfter	X			
notOnOrBefore	X			
notOnOrAfter	X			
notWithin	X			

Search Types (Table 2)				
Enumerations	SearchDate FieldOperator	SearchEnum MultiSelect FieldOperator	SearchMulti SelectField Operator	SearchDate
notEmpty	X			
lastBusinessWeek				X
lastFiscalQuarter				X
lastFiscalQuarterToDate				X
lastFiscalYear				X
lastFiscalYearToDate				X
lastMonth				X
lastMonthToDate				X
lastRollingQuarter				X
lastRollingYear				X
lastWeek				X
lastWeekToDate				X
nextBusinessWeek				X
nextFiscalQuarter				X
nextFiscalYear				X
nextFourWeeks				X
nextMonth				X
nextOneMonth				X
nextOneQuarter				X
nextOneWeek				X
nextOneYear				X
nextWeek				X
previousOneDay				X
previousOneMonth				X
previousOneQuarter				X
previousOneWeek				X
previousOneYear				X
previousRollingQuarter				X
previousRollingYear				X
sameMonthLastFiscalQuarter				X
sameMonthLastFiscalQuarterT oDate				X
sameMonthLastFiscalYear				X

Search Types (Table 2)				
Enumerations	SearchDate FieldOperator	SearchEnum MultiSelect FieldOperator	SearchMulti SelectField Operator	SearchDate
sameMonthLastFiscalYearToDate				X
sameQuarterLastFiscalYear				X
sameQuarterLastFiscalYearToDate				X
thisBusinessWeek				X
thisFiscalQuarter				X
thisFiscalQuarterToDate				X
thisFiscalYear				X
thisFiscalYearToDate				X
thisMonth				X
thisMonthToDate				X
thisRollingQuarter				X
thisRollingYear				X
thisWeek				X
thisWeekToDate				X
thisYear				X
today				X
tomorrow				X
yesterday				X

Chapter 5 Operations

This section describes the operations that can be used in requests to retrieve and change data in your NetSuite account. For example, you can use operations to login to your NetSuite account, and then retrieve, add, update or delete a record.

The following operations are supported:

- login
- mapSso
- logout
- addList
- add
- updateList
- update
- deleteList
- delete
- search
- searchMore
- searchNext
- getList
- get
- getAll
- getSelectValue
- getCustomization
- getItemAvailability
- attach / detach
- changePasswordOrEmail
- getDeleted
- initialize / initializeList

login

The login operation is used to authenticate a user and start a new Web services session in NetSuite. The login operation is similar to the login for the UI. The login operation provides a passport that includes a username, password, account and role. On success, the NetSuite server sets a cookie and establishes a session.



Note: All sessions have a time-out. If the session times out, the next operation fails. Web services requests initiated through a client application must have the ability to execute the login operation when a session has timed out and then submit the original request again. For more information on session management, see “[Session Management](#)” on page 42.

All available Web services operations require that the user first be logged in. If an operation is executed before a login is performed, it fails and the InvalidSessionFault is returned.

The login operation also verifies that the specified account has the Web Services feature turned on. If this is not the case, the SOAP fault InvalidCredentialsFault is returned with a code of WEB_SERVICES_NOT_ENABLED.

Request

The LoginRequest type is used for this request. It contains the following fields:

Element Name	XSD Type	Notes
passport	Passport	Contains all the required credentials including username, password, account and role to authenticate the user and create a new session.

The Passport type takes the following fields:

- 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 don't 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 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 relogin with a different role.

Element Name	XSD Type	Notes
userID	string	References an existing user record.
role	RecordRef	References an existing role record.
isDefault	boolean	If true, the role is set as the default role for this user.
isInactive	boolean	If true, this role is currently unavailable for the current user.

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 given endpoint.

```
String userid = getHeader(port, "sessionInfo");
public static String getHeader(NetSuitePortType port, String headerName) {
    com.netsuite.webservices.platform.NetSuiteBindingStub stub =
        (com.netsuite.webservices.platform.NetSuiteBindingStub) port;
    SOAPHeaderElement [] headers = stub.getResponseHeaders();
    for (int i=0; i< headers.length; i++)
    {
        SOAPHeaderElement header = headers[i];
        if (header.getName().equals(headerName))
        {
            Iterator childElements = header.getChildElements();
            while (childElements.hasNext())
            {
                SOAPElement el = (SOAPElement) childElements.next();
                return el.getValue();
            }
        }
    }
    return null;
}
```

Faults

This operation can throw one of the following faults. See “Soap Fault Status Codes” on page 152 for more information on faults.

- InsufficientPermissionFault
- InvalidAccountFault
- 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 internal IDs see “IDs Associated with Roles” on page 30.

```
<soap:Body>
  <platformMsgs:login>
    <platformMsgs:passport>
      <platformCore:email>jsmith@yahoo.com</platformCore:email>
      <platformCore:password>password123</platformCore:password>
      <platformCore:account>111111</platformCore:account>
      <platformCore:role internalId="3"/>
    </platformMsgs:passport>
  </platformMsgs:login>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
  <loginResponse xmlns="urn:messages_1_2.platform.webservices.netsuite.com">
    <sessionResponse xmlns="urn:messages_1_2.platform.webservices.netsuite.com">
      <ns1:status isSuccess="true"
xmlns:ns1="urn:core_1_2.platform.webservices.netsuite.com"/>
      <ns2:wsRoleList xmlns:ns2="urn:core_1_2.platform.webservices.netsuite.com">
        <ns2:wsRole>
          <ns2:role internalId="3">
            <ns2:name>Administrator</ns2:name>
          </ns2:role>
          <ns2:isDefault>>false</ns2:isDefault>
          <ns2:isInactive>>false</ns2:isInactive>
        </ns2:wsRole>
        <ns2:wsRole>
          <ns2:role internalId="15">
            <ns2:name>Employee Center</ns2:name>
          </ns2:role>
          <ns2:isDefault>>false</ns2:isDefault>
          <ns2:isInactive>>false</ns2:isInactive>
        </ns2:wsRole>
        <ns2:wsRole>
          <ns2:role internalId="22">
            <ns2:name>Intranet Manager</ns2:name>
          </ns2:role>
          <ns2:isDefault>>false</ns2:isDefault>
          <ns2:isInactive>>false</ns2:isInactive>
        </ns2:wsRole>
        <ns2:wsRole>
          <ns2:role internalId="25">
            <ns2:name>System Administrator</ns2:name>
          </ns2:role>
          <ns2:isDefault>>false</ns2:isDefault>
          <ns2:isInactive>>false</ns2:isInactive>
        </ns2:wsRole>
      </ns2:wsRoleList>
    </sessionResponse>
  </loginResponse>
</soapenv:Body>
```

C#

```

private void login( bool isAuto )
{
if ( !_isAuthenticated )
{
// Check whether this is a forced login as part of another operation
if ( isAuto )
_out.WriteLine( "\nYou need to first login before invoking this operation ..."
);

// Enable client cookie management. This is required.
_service.CookieContainer = new CookieContainer();

// Populate Passport object with all login information
Passport passport = new Passport();
RecordRef role = new RecordRef();

// Determine whether to get login information from config
// file or prompt for it
if ( "true".Equals( _dataCollection["promptForLogin"] ) )
{
_out.WriteLine( "\nPlease enter your login information: " );
_out.write( "E-mail: " );
passport.email = _out.readLn();
_out.write( "Password: " );
passport.password = _out.readLn();
_out.write( "Role nsKey (press enter for default administrator role): " );
role.internalId = _out.readLn();
passport.role =role;
_out.write( "Account: " );
passport.account = _out.readLn();
}
else
{
passport.email = _dataCollection["login.email"];
passport.password = _dataCollection["login.password"];
role.internalId = _dataCollection["login.roleNSkey"];
passport.role =role;
passport.account= _dataCollection["login.acct"];
}

// Login to NetSuite
_out.info( "\nLogging into NetSuite" );
_out.info( "Username: " + passport.email );
_out.info( "Account: " + passport.account );
Status status = (_service.login( passport )).status;

// Process response
if ( status.isSuccess == true )
{
_isAuthenticated = true;
_out.info( "\nThe user with nsKey=" + _service.sessionInfo.userId + " was
logged
in successful and a new session has been created." );
}
else
{
// Should never get here since any problems with the
// login should have resulted in a SOAP fault
_out.error( getStatusDetails( status ) );
}
}
}
}

```

Java

```
public void login(boolean isAuto) throws RemoteException {
    if (!_isAuthenticated) {
        // Check whether this is a forced login as part of another operation
        if (isAuto)
            _console
                .writeln("\nYou need to first login before invoking this operation ...");

        // Populate Passport object with all login information
        Passport passport = new Passport();
        RecordRef role = new RecordRef();

        // Determine whether to get login information from config
        // file or prompt for it
        if ("true".equals(_properties.getProperty("promptForLogin"))) {
            _console.writeln("\nPlease enter your login information: ");
            System.out.print("E-mail: ");
            passport.setEmail(_console.readLine());
            System.out.print("Password: ");
            passport.setPassword(_console.readLine());
            System.out.print("Role nsKey (press enter for default administrator role):
");
            role.setInternalId(_console.readLine());
            passport.setRole(role);
            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("\nLogging into NetSuite");
        _console.info("Username: " + passport.getEmail());
        _console.info("Account: " + passport.getAccount());

        Status status = (_port.login(passport)).getStatus();
        // Process the response
        if (status.isIsSuccess() == true) {
            _isAuthenticated = true;
            _console
                .info("\nThe login was successful and a new session has been created.");
        } else {
            // Should never get here since any problems with the
            // login should have resulted in a SOAP fault
            _console.error(getStatusDetails(status));
        }
    }
}
```

mapSso

Single Signon (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 once per session.

The mapSso operation streamlines the provisioning of Single Signon User Accounts by providing the ability to correlate a partner/customer specified internal ID with a Netsuite credential. This automates the mapping between external applications credentials and NetSuite's credentials for a user.



Important: This operation is a one time mapping — NOT a login. The operation does NOT include the ability to login via a Single Signon Token, nor does it allow for the provisioning of a partner — it does not aid with public/private key exchange. Also, this mapping is allowed ONLY for an administrator.

This section describes the mapSso operation. For more detailed information on NetSuite's SSO technology, contact your NetSuite Account Manager or Professional Services Manager for the **Single Signon Developer's Guide**.

Request

The MapSsoRequest type is used for this request. It contains the following fields:

Element Name	XSD Type	Notes
ssoCredentials	SsoCredentials	Contains all the required credentials including username, password, account and role to authenticate the user and create a new session.

The SsoCredentials type requires the following fields:

- email
- password
- account
- role
- authentication Token: this is a string representing the Private Key encrypted Netsuite token.

This token, prior to any hex-encoding, is of the form:

```
<companyID><space><userID><space><timestamp>
```

<companyId> is the integration partner's affiliate ID as provided to the partner from NetSuite (typically five numeric digits long). The <companyId> should be equal to <userId> when the sending system does not support the notion of subusers. Since spaces are used to delimit subtokens, no spaces can be used in the tokens. The timestamp string is a decimal representation of the number of milliseconds since January 1, 1970, 00:00:00 GMT. Optionally, a return URL can also be included. This is the URL the user is redirected to upon logout from NetSuite.

- partnerId



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.

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](#)” on [page 152](#) for more information on faults.

- InsufficientPermissionFault
- InvalidAccountFault
- InvalidAccountFault
- InvalidCredentialsFault
- InvalidVersionFault
- UnexpectedErrorFault

Sample Code: SOAP

```
<soap:Body>
<platformMsgs:mapSso>
<platformMsgs:ssoCredentials>
<platformCore:email>bbrown@yahoo.com</platformCore:email>
<platformCore:password>Cracker1</platformCore:password>
<platformCore:account>TSTDVR211915</platformCore:account>
<platformCore:authenticationToken>[enter token here]</
platformCore:authenticationToken>
<platformCore:partnerId>[enter partner id here]</platformCore:partnerId>
</platformMsgs:ssoCredentials>
</platformMsgs:mapSso>
</soap:Body>
```

Sample Code: 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("TSTDVR00000");
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());
    }

```

logout

The logout operation is used to terminate an active session.



Note: If you explicitly logout 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](#)” on [page 152](#) for more information on faults.

- UnexpectedErrorFault

Sample Code

Following is an example that contains an excerpt of the SOAP body for both the request and response.

SOAP Request

```
<logout/>
```

SOAP Response

```

<logoutResponse>
  <status isSuccess="true"/>
</logoutResponse>

```

C#

```

private void logout()
{
    if ( _isAuthenticated )
    {
        _out.info( "\nLogging out of NetSuite\n" );

        // Logout from NetSuite
        Status status = ( _service.logout() ).status;

        if ( status.isSuccess == true )
        {
            _isAuthenticated = false;
            _out.info( "Logout successful" );
        }
        else
        {

```

```

        // Should never get here since any problems with the
        // logout should have resulted in a SOAP fault
        _out.error( getStatusDetails( status ) );
    }
}
else
{
    _out.info(
        "\nThe logout() operation cannot be invoked because there is no active
session. " +
        "You must be first logged on before attempting to logout.\n" );
}
}
}

```

Java

```

public void logout() throws RemoteException {
    if (_isAuthenticated) {
        _console.info("\nLogging out of NetSuite\n");

        // Logout from NetSuite
        Status status = (_port.logout()).getStatus();

        if (status.isIsSuccess() == true) {
            _isAuthenticated = false;
            _console.info("Logout successful");
        } else {
            // Should never get here since any problems with the
            // logout should have resulted in a SOAP fault
            _console.error(getStatusDetails(status));
        }
    } else {
        _console
        .info("\nThe logout() operation cannot be invoked because there is no active
session. "
        + "You must be first logged on before attempting to logout.\n");
    }
}
}

```

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

Request

The AddListRequest type is used for the request.

Element Name	XSD Type	Notes
record	Record	Contains an array of record objects. The record type is an abstract type so an instance of a type that extends record must be used—such as Customer or Event.

Response

The AddListResponse type is used for the response.

Element Name	XSD Type	Notes
response	WriteResponse	Contains an array of WriteResponse objects, each of which contains details on the status of that add operation and a reference to that created record.

Faults

This operation can throw one of the following faults. See “[Soap Fault Status Codes](#)” on [page 152](#) for more information on faults.

- InvalidSessionFault
- ExceededRateLimitFault
- ExceededMaxRecordsFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In the following example, two customer records are added. When using the addList operation you can submit two different record types in the same request.

```
<soap:Body>
  <platformMsgs:addList>
    <platformMsgs:record xsi:type="listRel:Customer">
      <listRel:entityId>Shutter Fly</listRel:entityId>
      <listRel:companyName>Shutter Fly, Inc</listRel:companyName>
    </platformMsgs:record>
    <platformMsgs:record xsi:type="listRel:Customer">
      <listRel:entityId>GNC</listRel:entityId>
      <listRel:companyName>GNC Corp</listRel:companyName>
    </platformMsgs:record>
  </platformMsgs:addList>
</soap:Body>
```

SOAP Response

In the response, notice that the internalID for each record added is returned.

```

<soapenv:Body>
<addListResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
<writeResponseList xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
  <writeResponse>
    <ns1:status isSuccess="true"
xmlns:ns1="urn:core_2_6.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_2_6.platform.webservices.netsuite.com"/>
  </writeResponse>
<writeResponse>
  <ns3:status isSuccess="true"
xmlns:ns3="urn:core_2_6.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_2_6.platform.webservices.netsuite.com"/>
  </writeResponse>
</writeResponseList>
</addListResponse>
</soapenv:Body>

```

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.

Request

The AddRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
record	Record	The record type is an abstract type so an instance of a type that extends record must be used — such as Customer or Event.

Response

The AddResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
response	WriteResponse	Contains details on the status of the operation and a reference to the created record.

Faults

This operation can throw one of the following faults. See “Soap Fault Status Codes” on page 152 for more information on faults.

- InvalidSessionFault
- ExceededRateLimitFault
- ExceededMaxRecordsFault

- UnexpectedErrorFault

Sample Code

SOAP Request

In this example, a single customer record is added.

```
<soap:Body>
<platformMsgs:add>
  <platformMsgs:record xsi:type="listRel:Customer">
    <listRel:entityId>Shutter Fly</listRel:entityId>
    <listRel:companyName>Shutter Fly, Inc</listRel:companyName>
  </platformMsgs:record>
</platformMsgs:add>
</soap:Body>
```

SOAP Response

In the response, notice that the internalID for the record added is returned.

```
<soapenv:Body>
<addResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
<writeResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
  <ns1:status isSuccess="true"
xmlns:ns1="urn:core_2_6.platform.webservices.netsuite.com"/>
  <baseRef internalId="979" type="customer" xsi:type="ns2:RecordRef"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns2="urn:core_2_6.platform.webservices.netsuite.com"/>
</writeResponse>
</addResponse>
</soapenv:Body>
```

C#

```
private void addCustomer()
{
  // This operation requires a valid session
  this.login( true );

  Customer customer = new Customer();

  // Set entityId, company name, and email
  if ( "true".Equals( _dataCollection["promptForFieldValues"] ) )
  {
    _out.WriteLine(
      "\nPlease enter the following customer information. " +
      "Note that some fields have already been populated. " );
    _out.write( "Entity name: " );
    customer.entityId = _out.readLn();
    _out.write( "Company name: " );
    customer.companyName = _out.readLn();
    _out.write( "E-mail: " );
    customer.email = _out.readLn();
    _out.write( "Sales Rep key: " );
    RecordRef salesRep = new RecordRef();
    salesRep.internalId = _out.readLn();
    //salesRep.type = RecordType.contact;
    //salesRep.typeSpecified = true;
    customer.salesRep = salesRep;
  }
  else
  {
    customer.entityId = "XYZ Inc";
  }
}
```

```

        customer.companyName = "XYZ, Inc.";
        customer.email = "bsanders@yahoo.com";
    }

    // Set email preference
    customer.emailPreference = CustomerEmailPreference._html;
    customer.emailPreferenceSpecified = true;

    // Set entity status. The nsKey can be obtained from Setup > SFA > Customer
    Statuses.
    // The default status is "Closed Won" which has an nsKey of 13
    RecordRef status = new RecordRef();
    if ( "true".Equals( _dataCollection["promptForFieldValues"] ) )
    {
        _out.write( "Entity status nsKey (press enter for default value of Closed
Won): " );
        String statusKey = _out.readLine();
        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.addressee = "William Sanders";
    address.attention = "William Sanders";
    address.addr1 = "4765 Sunset Blvd";
    address.city = "San Francisco";
    address.state = State._california;
    address.zip = "94131";
    address.country = Country._unitedStates;

    // Attach the CustomerAddressbookList to the customer
    CustomerAddressbookList addressList = new CustomerAddressbookList();
    CustomerAddressbook[] addresses = new CustomerAddressbook[1];
    addresses[0] = address;
    addressList.addressbook = addresses;
    customer.addressbookList = addressList;

    // Invoke add() operation
    WriteResponse response = _service.add( customer );

    // Print the document id from the SOAP header
    // out.info(
    // "\n\nThe add() operation with document id " + _service.documentInfo.nsID + " was
    processed " );

```

```

// Process the response
if ( response.status.isSuccess )
{
    _out.info(
        "\nThe following customer was added successfully:" +
        "\nkey=" + ((RecordRef) response.baseRef).internalId +
        "\nentityId=" + customer.entityId +
        "\ncompanyName=" + customer.companyName +
        "\nemail=" + customer.email +
        "\nstatusKey=" + customer.entityStatus.internalId +
        "\naddressbookList[0].label=" + customer.addressbookList.addressbook[0].label
    );

    // Create a second customer that's a sub-customer of the first customer
    /*
    Customer subCustomer = new Customer();

    subCustomer.entityId = "XYZ Japan";

    // Set the parent customer
    recordRef = new RecordRef();
    recordRef.internalId = ((RecordRef) response.baseRef).internalId;
    //recordRef.internalId = "125";
    subCustomer.parent = recordRef;

    // Invoke add() operation
    response = _service.add( subCustomer );

    // Process the response
    //"\nThe add() operation with job id " + _service.sessionInfo.jobID + " was
processed " );
    if ( !response.status.isSuccess )
    {
        _out.info( "\nThe customer was not added:" );
        _out.error( getStatusDetails( response.status ) );
    }
    else
    {
        _out.info(
            "\nThe following customer was added successfully:" +
            "\nkey=" + ((RecordRef) response.baseRef).internalId +
            "\nentityId=" + customer.entityId +
            "\ncompanyName=" + customer.companyName);
        }
    }
    */
}
else
{
    _out.error( "The customer was not added:", true );
    _out.error( getStatusDetails( response.status ) );
}
}
}

```

Java

```

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

    Customer customer = new Customer();

    // Set entityId, company name, and email
    if ("true".equals(_properties.getProperty("promptForFieldValues"))) {

```

```
_console
.writeLn("\nPlease enter the following customer information. "
+ "Note that some fields have already been populated. ");
_console.write("Entity name: ");
customer.setEntityId(_console.readLn());
_console.write("Company name: ");
customer.setCompanyName(_console.readLn());
_console.write("E-mail: ");
customer.setEmail(_console.readLn());
} else {
customer.setEntityId("XYZ Inc");
customer.setCompanyName("XYZ, Inc.");
customer.setEmail("bsanders@yahoo.com");
}

// Set email preference.
customer.setEmailPreference(CustomerEmailPreference._hTML);

// Set entity status. The nsKey can be obtained from Setup > SFA >
// Customer Statuses.
// The default status is "Closed Won" which has an nsKey of 13
RecordRef status = new RecordRef();
if ("true".equals(_properties.getProperty("promptForFieldValues"))) {
_console
.write("Entity status nsKey (press enter for default value of Closed Won): ");
String statusKey = _console.readLn();
if (statusKey.equals("")) {
status.setInternalId("13");
} else {
status.setInternalId(statusKey);
}
} else {
status.setInternalId("13");
}

customer.setEntityStatus(status);

// Populate the address list for this customer. You can put in as many
// addresses as you like.
CustomerAddressbook address = new CustomerAddressbook();
address.setDefaultShipping(Boolean.TRUE);
address.setDefaultBilling(Boolean.FALSE);
address.setLabel("Shipping Address");
address.setAddressee("William Sanders");
address.setAttention("William Sanders");
address.setAddr1("4765 Sunset Blvd");
address.setCity("San Francisco");
address.setState(State._california);
address.setZip("94131");
address.setCountry(Country._unitedStates);

// Attach the CustomerAddressbookList to the customer
CustomerAddressbookList addressList = new CustomerAddressbookList();
CustomerAddressbook[] addresses = new CustomerAddressbook[1];
addresses[0] = address;
addressList.setAddressbook(addresses);
customer.setAddressbookList(addressList);

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

// Print the document id from the SOAP header
// _console.info(
// "\nThe add() operation with document id " + _port.documentInfo.nsID +
```

```

// " was processed " );

// Process the response
if (response.getStatus().isIsSuccess()) {
    _console.info("\nThe following customer was added successfully:"
+ "\nkey="
+ ((RecordRef) response.getBaseRef()).getInternalId()
+ "\nentityId="
+ customer.getEntityId()
+ "\ncompanyName="
+ customer.getCompanyName()
+ "\nemail="
+ customer.getEmail()
+ "\nstatusKey="
+ customer.getEntityStatus().getInternalId()
+ "\naddressbookList[0].label="
+ customer.getAddressbookList().getAddressbook(0)
.getLabel());
} else {
    _console.error("The customer was not added:", true);
    _console.error(getStatusDetails(response.getStatus()));
}
}
}

```

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.



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 [“Working with Hidden Fields”](#) on page 35.

Request

The UpdateListRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
record	Record	Contains an array of record objects. The record type is an abstract type so an instance of a type that extends record must be used—such as Customer or Event.

Response

The UpdateListResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
response	WriteResponse	Contains an array of WriteResponse objects, each of which contains details on the status of that update operation and a reference to the created record.

Faults

This operation can throw one of the following faults. See “Soap Fault Status Codes” on page 152 for more information on faults.

- InvalidSessionFault
- ExceededRateLimitFault
- ExceededMaxRecordsFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In the following example, two customer records are updated. The first has the single field `companyName` updated, while the second updates two fields, `entityID` and `companyName`. The `internalID` for each record must be provided and the type of record (`Customer`) to be updated.

```
<soap:Body>
<platformMsgs:updateList>
  <platformMsgs:record internalId="980" xsi:type="listRel:Customer">
    <listRel:companyName>Shutter Fly Corporation</listRel:companyName>
  </platformMsgs:record>
  <platformMsgs:record internalId="981" xsi:type="listRel:Customer">
    <listRel:entityId>GNCC</listRel:entityId>
    <listRel:companyName>GNCC Corp</listRel:companyName>
  </platformMsgs:record>
</platformMsgs:updateList>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
<updateListResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
<writeResponseList xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
  <writeResponse>
    <ns1:status isSuccess="true"
xmlns:ns1="urn:core_2_6.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_2_6.platform.webservices.netsuite.com"/>
  </writeResponse>
  <writeResponse>
    <ns3:status isSuccess="true"
xmlns:ns3="urn:core_2_6.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_2_6.platform.webservices.netsuite.com"/>
  </writeResponse>
</writeResponseList>
```

```
</updateListResponse>
</soapenv:Body>
```

C#

```
private void updateCustomerList()
{
    // This operation requires a valid session
    this.login( true );

    // Prompt for list of nsKeys and put in an array
    _out.write( "\nEnter nsKeys for customer records to be updated (separated by
commas): " );
    String reqKeys = _out.readLine();
    string [] nsKeys = reqKeys.Split( new Char[] {','} );

    // Create an array of Record objects to hold the customers
    Record[] records = new Record[nsKeys.Length];

    // For each submitted nsKey, populate a customer object
    for ( int i=0; i<nsKeys.Length; i++)
    {
        Customer customer = new Customer();

        // Update name
        customer.entityId = "XYZ Inc " + i;
        customer.companyName = "XYZ, Inc. " + i;

        customer.internalId = nsKeys[i].Trim();
        records[i] = customer;
    }

    // Invoke updateList() operation to update customers
    WriteResponse[] responses = _service.updateList( records );

    // Process responses for all successful updates
    _out.info( "\nThe following customers were updated successfully:" );
    bool hasFailures = false;
    for ( int i=0; i<responses.Length; i++ )
    {
        if ( (responses[i] != null) && (responses[i].status.isSuccess) )
        {
            _out.info( "\nCustomer[" + i + "]:" );
            _out.info(
                "key=" + ((RecordRef) responses[i].baseRef).internalId +
                "\nentityId=" + ((Customer) records[i]).entityId +
                "\ncompanyName=" + ((Customer) records[i]).companyName );
        }
        else
        {
            hasFailures = true;
        }
    }

    // Process responses for all unsuccessful updates
    if ( hasFailures )
    {
        _out.info( "\nThe following customers were not updated:\n" );
        for ( int i=0; i<responses.Length; i++ )
        {
            if ( (responses[i] != null) && (!responses[i].status.isSuccess) )
            {
                _out.info( "Customer[" + i + "]:" );
                _out.info( "key=" + ((RecordRef) responses[i].baseRef).internalId );
                _out.errorForRecord( getStatusDetails( responses[i].status ) );
            }
        }
    }
}
```

```

    }
  }
}

```

Java

```

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

    // Prompt for list of nsKeys and put in an array
    _console
.write("\nEnter nsKeys for customer records to be updated (separated by commas):
");
String reqKeys = _console.readLine();
String[] nsKeys = reqKeys.split(",");

    // Create an array of Record objects to hold the customers
Record[] records = new Record[nsKeys.length];

    // For each submitted nsKey, populate a customer object
for (int i = 0; i < nsKeys.length; i++) {
    Customer customer = new Customer();

    // Update name
customer.setEntityId("XYZ Inc " + i);
customer.setCompanyName("XYZ, Inc. " + i);

    customer.setInternalId(nsKeys[i].trim());
records[i] = customer;
}

    // Invoke updateList() operation to update customers
WriteResponseList responseList = _port.updateList(records);

    // Process responses for all successful updates
WriteResponse[] responses = responseList.getWriteResponse();
boolean hasFailures = false;
_console.info("\nThe following customers were updated successfully:");
for (int i = 0; i < responses.length; i++) {
    if ((responses[i] != null)
    && (responses[i].getStatus().isIsSuccess())) {
        _console.info("\nCustomer[" + i + "]:");
        _console.info("key="
+ ((RecordRef) responses[i].getBaseRef()).getInternalId()
+ "\nentityId="
+ ((Customer) records[i]).getEntityId()
+ "\ncompanyName="
+ ((Customer) records[i]).getCompanyName());
    } else {
        hasFailures = true;
    }
}

    // Process responses for all unsuccessful updates
if (hasFailures) {
_console.info("\nThe following customers were not updated:\n");
for (int i = 0; i < responses.length; i++) {
    if ((responses[i] != null)
    && (!responses[i].getStatus().isIsSuccess())) {
        _console.info("Customer[" + i + "]:");
        _console.info("key="

```

```

        + ((RecordRef) responses[i].getBaseRef()).getInternalId());
        _console.errorForRecord(getStatusDetails(responses[i]
        .getStatus()));
    }
}
}
}

```

update

The update operation is used to update an instance of a record in NetSuite. It is similar to the updateList operation but only allows one record to be updated 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, refer to [“Working with Hidden Fields” on page 35](#). Also, custom fields can only be set to NULL by submitting the field in nullFieldList. For more information, refer to [“CustomFieldList” on page 13](#) of the Records Guide.

In order to ensure that the most recent data for a given 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.

Request

The UpdateRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
record	Record	Contains an array of record objects. The record type is an abstract type so an instance of a type that extends record must be used—such as Customer or Event.

Response

The UpdateResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
response	WriteResponse	Contains details on the status of the operation and a reference to the updated record.

Faults

This operation can throw one of the following faults. See “Soap Fault Status Codes” on page 152 for more information on faults.

- InvalidSessionFault
- ExceededRateLimitFault
- ExceededMaxRecordsFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In the following example, a customer’s companyName is updated. The internal ID for the customer must be provided in the request.

```
<soap:Body>
<platformMsgs:update>
  <platformMsgs:record internalId="980" xsi:type="listRel:Customer">
    <listRel:companyName>Shutter Fly Corporation</listRel:companyName>
  </platformMsgs:record>
</platformMsgs:update>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
<updateResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
<writeResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
  <ns1:status isSuccess="true"
xmlns:ns1="urn:core_2_6.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_2_6.platform.webservices.netsuite.com"/>
</writeResponse>
</updateResponse>
</soapenv:Body>
```

C#

```
private void updateCustomer()
{
  // This operation requires a valid session
  this.login( true );

  Customer customer = new Customer();

  // Get nsKey for update
  _out.write( "\nEnter nsKey for customer record to be updated : " );
  customer.internalId = _out.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 = State._california;
address.country = Country._unitedStates;

// Attach the address to the customer
CustomerAddressbookList addressList = new CustomerAddressbookList();
CustomerAddressbook[] addresses = new CustomerAddressbook[1];
addresses[0] = address;
addressList.addressbook = addresses;
customer.addressbookList = addressList;

// Invoke add() operation
WriteResponse response = _service.update( customer );

// Process the response
if ( response.status.isSuccess )
{
    _out.info(
        "\nThe following customer was updated successfully:" +
        "\nkey=" + ((RecordRef) response.baseRef).internalId +
        "\nentityId=" + customer.entityId +
        "\ncompanyName=" + customer.companyName +
        "\nemail=" + customer.email +
        "\naddressbookList[0].label=" + customer.addressbookList.addressbook[0].label
    );
}
else
{
    _out.error( getStatusDetails( response.status ) );
}
}

```

Java

```

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

    Customer customer = new Customer();

    // Get nsKey for update
    _console.write("\nEnter nsKey for customer record to be updated : ");
    customer.setInternalId(_console.readLine().toUpperCase());

    // Set name and email
    customer.setEntityId("XYZ 2 Inc");
    customer.setCompanyName("XYZ 2, Inc.");
    customer.setEmail("bsanders@xyz.com");

    // Populate the address. Updating a list through WS results in the
    // entire contents of the previous list being replaced by the new
    // list.
    CustomerAddressbook address = new CustomerAddressbook();
    address.setDefaultBilling(Boolean.TRUE);
    address.setDefaultShipping(Boolean.FALSE);
    address.setLabel("Billing Address");
    address.setAddr1("4765 Sunset Blvd");
}

```

```

address.setCity("San Mateo");
address.setState(State._california);
address.setCountry(Country._unitedStates);

// Attach the address to the customer
CustomerAddressbookList addressList = new CustomerAddressbookList();
CustomerAddressbook[] addresses = new CustomerAddressbook[1];
addresses[0] = address;
addressList.setAddressbook(addresses);
customer.setAddressbookList(addressList);

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

// Process the response
if (response.getStatus().isIsSuccess()) {
    _console.info("\nThe following customer was updated successfully:"
        + "\nkey="
        + ((RecordRef) response.getBaseRef()).getInternalId()
        + "\nentityId="
        + customer.getEntityId()
        + "\ncompanyName="
        + customer.getCompanyName()
        + "\nemail="
        + customer.getEmail()
        + "\naddressbookList[0].label="
        + customer.getAddressbookList().getAddressbook(0)
        .getLabel());
} else {
    _console.error(getStatusDetails(response.getStatus()));
}
}

```

Updating Record Lists

When updating a list of records (a sublist) within a business record you can NOT update a specific item in the list. Instead you must interact with the sublist as a whole. For details, refer to “Working with Sublists” on page 32.

deleteList

The deleteList operations is used to delete one or more existing instances of a certain 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 delete a customer and a contact within a single request using this operation.

Request

The DeleteListRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
record	RecordRef	Must contain a reference to an existing instance of a record.

Response

The DeleteListResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
response	WriteResponse	Contains an array of WriteResponse objects, each of which contains details on the status of the delete operation and a reference to the deleted record.

Faults

This operation can throw one of the following faults. See “Soap Fault Status Codes” on page 152 for more information on faults.

- InvalidSessionFault
- ExceededRateLimitFault
- ExceededMaxRecordsFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In the following example, two customer records are deleted. You can delete records of different types in the same request.

```
<soap:Body>
<platformMsgs:deleteList>
  <platformMsgs:baseRef internalId="981" type="customer"
xsi:type="platformCore:RecordRef"/>
  <platformMsgs:baseRef internalId="982" type="customer"
xsi:type="platformCore:RecordRef"/>
</platformMsgs:deleteList>
</soap:Body>
```

SOAP Response

Note that the status for each item in the request is returned. If a single item in the list errors, the other records submitted will still be processed.

```
<soapenv:Body>
<deleteListResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
<writeResponseList xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
  <writeResponse>
    <ns1:status isSuccess="true"
xmlns:ns1="urn:core_2_6.platform.webservices.netsuite.com"/>
    <baseRef internalId="981" type="customer" xsi:type="ns2:RecordRef"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns2="urn:core_2_6.platform.webservices.netsuite.com"/>
  </writeResponse>
  <writeResponse>
    <ns3:status isSuccess="true"
xmlns:ns3="urn:core_2_6.platform.webservices.netsuite.com"/>
    <baseRef internalId="982" type="customer" xsi:type="ns4:RecordRef"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns4="urn:core_2_6.platform.webservices.netsuite.com"/>
  </writeResponse>
</writeResponseList>
</deleteListResponse>
```

```

</writeResponseList>
</deleteListResponse>
</soapenv:Body>

```

C#

```

private void deleteCustomerList()
{
    // This operation requires a valid session
    this.login( true );

    // Prompt for list of nsKeys and put in an array
    _out.write( "\nEnter nsKeys for customer records to be deleted (separated by
commas): " );
    String reqKeys = _out.readLine();
    string [] nsKeys = reqKeys.Split( new Char[] {','} );

    // First get the records from NS
    _out.write( "\nChecking validity of nsKeys by using getList() to retrieve
records ...\n" );
    int numRecords = getCustomerList( nsKeys, true );

    // Delete records, but only if there are records to delete
    if ( numRecords > 0 )
    {
        // Build an array of RecordRef objects
        RecordRef[] recordRefs = new RecordRef[ nsKeys.Length ];
        for (int i=0; i<nsKeys.Length; i++)
        {
            RecordRef recordRef = new RecordRef();
            recordRef.internalId = nsKeys[i];
            recordRefs[i] = recordRef;
            recordRefs[i].type = RecordType.customer;
            recordRefs[i].typeSpecified = true;
        }

        System.Console.Out.Write( "\nDelete all the records above? [Y/N]:" );
        String userResponse = _out.readLine().ToUpper();
        System.Console.Out.Write( "\n" );

        // Delete records
        if ( userResponse.Equals( "Y" ) )
        {
            WriteResponse[] delResponses = _service.deleteList( recordRefs );

            // process response
            _out.info( "\nThe following customers were deleted:\n" );
            for (int i=0; i<delResponses.Length; i++ )
            {
                if ( delResponses[i].status.isSuccess )
                {
                    _out.info( "Customer[" + i + "]:" );
                    _out.info( "key=" + ((RecordRef) delResponses[i].baseRef).internalId
);
                }
                else
                {
                    _out.info( "Customer[" + i + "]:" );
                    _out.errorForRecord( getStatusDetails( delResponses[i].status ) );
                }
            }
        }
        else
        {
            _out.info( "\nSince your answer was not Y, the records were not deleted." );
        }
    }
}

```

```

    }
  }
  else
  {
    _out.info( "\nThere were no valid records to be deleted." );
  }
}

```

Java

```

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

  // Prompt for list of nsKeys and put in an array
  _console
  .write("\nEnter nsKeys for customer records to be deleted (separated by commas):
");
  String reqKeys = _console.readLine();
  String[] nsKeys = reqKeys.split(",");

  // First get the records from NS
  _console
  .write("\nChecking validity of nsKeys by using getList() to retrieve records
...\n");
  int numRecords = getCustomerList(nsKeys, true);

  // Delete records, but only if there are records to delete
  if (numRecords > 0) {
    // Build an array of RecordRef objects
    RecordRef[] recordRefs = new RecordRef[nsKeys.length];
    for (int i = 0; i < nsKeys.length; i++) {
      RecordRef recordRef = new RecordRef();
      recordRef.setInternalId(nsKeys[i]);
      recordRefs[i] = recordRef;
      recordRefs[i].setType(RecordType.customer);
    }

    _console.write("\nDelete all the records above? [Y/N]:");
    String userResponse = _console.readLine().toUpperCase();
    _console.write("\n");

    // Delete records
    if (userResponse.equals("Y")) {
      WriteResponseList delResponseList = _port
      .deleteList(recordRefs);

      // process response
      WriteResponse[] delResponses = delResponseList
      .getWriteResponse();
      _console.info("\nThe following customers were deleted:\n");
      for (int i = 0; i < delResponses.length; i++) {
        if (delResponses[i].getStatus().isIsSuccess()) {
          _console.info("Customer[" + i + "]:");
          _console.info("key="
          + ((RecordRef) delResponses[i].getBaseRef())
          .getInternalId());
        } else {
          _console.info("Customer[" + i + "]:");
          _console
          .errorForRecord(getStatusDetails(delResponses[i]
          .getStatus()));
        }
      }
    }
  }
}

```

```

    }
  } else {
    _console
      .info("\nSince your answer was not Y, the records were not deleted.");
  }
} else {
  _console.info("\nThere were no valid records to be deleted.");
}
}
}

```

delete

The delete operation is used to delete an existing instance of a record in NetSuite. It is identical in terms of functionality to the deleteList operation but only allows one record to be deleted per request.

Request

The DeleteRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
record	RecordRef	Must contain a reference to an existing instance of a record.

Response

The DeleteResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
response	WriteResponse	Contains details on the status of the delete operation and a reference to the deleted record.

Faults

This operation can throw one of the following faults. See “Soap Fault Status Codes” on [page 152](#) for more information on faults.

- InvalidSessionFault
- ExceededRateLimitFault
- ExceededMaxRecordsFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In this example, a single customer record is deleted.

```

<soap:Body>
  <platformMsgs:delete>
    <platformMsgs:baseRef internalId="980" type="customer"
      xsi:type="platformCore:RecordRef"/>
  </platformMsgs:delete>
</soap:Body>

```

```
</platformMsgs:delete>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
<deleteResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
<writeResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
  <ns1:status isSuccess="true"
xmlns:ns1="urn:core_2_6.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_2_6.platform.webservices.netsuite.com"/>
</writeResponse>
</deleteResponse>
</soapenv:Body>
```

C#

```
private void deleteCustomRecord()
{
    // This operation requires a valid session
    this.login( true );
    CustomRecordRef customRecordRef = new CustomRecordRef();
    //Prompt user for the nsKey for Custom Record type to be deleted
    _out.write( "Enter nsKey for Custom Record type to be deleted: " );
    customRecordRef.typeId = _out.readLn();
    //Prompt user for nsKey for Custom Record to be deleted
    _out.write( "Enter nsKey for Custom Record to be deleted: " );
    customRecordRef.internalId = _out.readLn();
    System.Console.Out.Write( "Delete the record above? [Y/N]:" );
    String userResponse = _out.readLn().ToUpper();
    // Delete records
    if ( userResponse.Equals( "Y" ) )
    {
        WriteResponse delResponse = _service.delete( customRecordRef );

        // process response
        _out.info( "\nThe following Custom Record deleted:\n" );
        if ( delResponse.status.isSuccess )
        {
            _out.info( "key=" + ((CustomRecordRef) delResponse.baseRef).internalId );
        }
        else
        {
            _out.errorForRecord( getStatusDetails( delResponse.status ) );
        }
    }
    else
    {
        _out.info( "\nSince your answer was not Y, the records were not deleted." );
    }
}
}
```

Java

```
public void deleteCustomRecord() throws RemoteException,
ExceededUsageLimitFault, UnexpectedErrorFault, InvalidSessionFault,
ExceededRecordCountFault {
    // This operation requires a valid session
    this.login(true);
    CustomRecordRef customRecordRef = new CustomRecordRef();
    // Prompt user for the nsKey for Custom Record type to be deleted
    _console.write("Enter nsKey for Custom Record type to be deleted: ");
    customRecordRef.setTypeId(_console.readLn());
    // Prompt user for nsKey for Custom Record to be deleted
    _console.write("Enter nsKey for Custom Record to be deleted: ");
```

```

customRecordRef.setInternalId(_console.readLine());
// customRecordRef.typeSpecified = true;
_console.write("Delete the record above? [Y/N]:");
String userResponse = _console.readLine().toUpperCase();
// Delete records
if (userResponse.equals("Y")) {
    WriteResponse delResponse = _port.delete(customRecordRef);

    // process response
    _console.info("\nThe following Custom Record deleted:\n");
    if (delResponse.getStatus().isIsSuccess()) {
        _console
            .info("key="
                + ((CustomRecordRef) delResponse.getBaseRef())
                .getInternalId());
    } else {
        _console.errorForRecord(getStatusDetails(delResponse
            .getStatus()));
    }
} else {
    _console
        .info("\nSince your answer was not Y, the records were not deleted.");
}
}
}

```

search

The search operation is used to execute a search on a specific record type based on a set of search criteria that is specified through a custom search object type `SearchRecord`. You can perform a search on a record type for fields specific for that record type only or you can perform a joined search based on fields specific for that record and filtered by any search fields on an associated record.

Working with Searches and Joined Searches

The `SearchRecord` type consists of a `Basic` element which references all of the search fields specific to that record type and a list of join elements that reference fields available from other exposed records.

For example, the `CustomerSearch` record consists of one `CustomerSearchBasic` element and seven join search elements: `parentCustomerJoin`, `subCustomerJoin`, `contactJoin`, `salesRepJoin`, `caseJoin`, `contactPrimaryJoin`, and `webSiteItemJoin`. Note that all `Basic` search types physically reside in the `platformCommon` XSD.

```

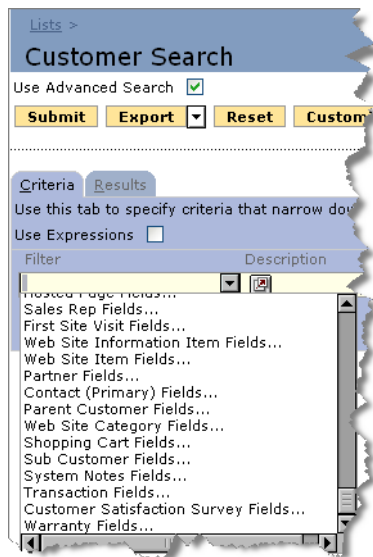
<complexType name="CustomerSearch">
  <complexContent>
    <extension base="platformCore:SearchRecord">
      <sequence>
        <element name="basic" type="platformCommon:CustomerSearchBasic"
          minOccurs="0"/>
        <element name="parentCustomerJoin"
          type="platformCommon:CustomerSearchBasic" minOccurs="0"/>
        <element name="subCustomerJoin"
          type="platformCommon:CustomerSearchBasic" minOccurs="0"/>
        <element name="contactJoin" type="platformCommon:ContactSearchBasic"
          minOccurs="0"/>
        <element name="salesRepJoin" type="platformCommon:EmployeeSearchBasic"
          minOccurs="0"/>
        <element name="caseJoin" type="platformCommon:SupportCaseSearchBasic"

```

```
        minOccurs="0"/>
        <element name="contactPrimaryJoin"
            type="platformCommon:ContactSearchBasic" minOccurs="0"/>
        <element name="webSiteItemJoin" type="platformCommon:ItemSearchBasic"
            minOccurs="0"/>
    </sequence>
</extension>
</complexContent>
</complexType>
```



Note: Only fields from **currently exposed** records can be specified as filter criteria for a joined search request. You can identify which records can be used as filter criteria in the NetSuite UI by navigating to the Search Record for the desired record type, enabling **Use Advanced Search** and then scrolling through the Filter drop-down list. Joined search records are indicated by the Record Name followed by an ellipsis (...). Search fields from any of the records listed here that are also currently exposed can be included in the search criteria.



Available Joined Searches

The following table outlines the currently available joined searches for each Search Record type.

Search Type	Available Joins
SupportCaseSearch	contactJoin customerJoin employeeJoin itemJoin
ContactSearch	caseJoin customerJoin customerPrimaryJoin opportunityJoin
CustomerSearch	supportCaseJoin contactJoin contactPrimaryJoin parentCustomerJoin salesRepJoin subCustomerJoin webSiteItemJoin
EmployeeSearch	transactionJoin
CalendarEventSearch	opportunityJoin
ItemSearch	shopperJoin
opportunitySearch	customerJoin decisionMakerJoin calendarEventJoin phoneCallJoin salesRepJoin taskJoin
PhoneCallSearch	contactJoin employeeJoin opportunityJoin
TaskSearch	contactJoin employeeJoin opportunityJoin
TransactionSearch	contactPrimaryJoin customerJoin employeeJoin itemJoin salesRepJoin



Note: In order to simplify code, the extension base is `platformCore:SearchRecord` for both the basic search element types and the regular search element types. For example, both `search(CustomerSearch)` object and `search(CustomerSearchBasic)` object are valid search operations.

Maintaining Code

The current version of the WSDL has updated the naming of search objects in order to accommodate the joined search functionality. In order to maintain backwards compatibility the following options are available:

- NO code change is required if you do NOT want to use the latest endpoint
- If you DO want to use the latest endpoint but do not need to take the advantage of joined searches yet you can just modify the search record you used to instantiate from `<Record>Search` to `<Record>SearchBasic`. This should be sufficient.
- If you want to use the latest endpoint and use the joined searches you can instantiate the search record as the previous bullet describes and add any joined searches

Returning Associated Lists

Using the 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 given list of customers. In order 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.

In the following example, all contacts associated with customers of `internalId` 1, 2 and 3 are returned.

Java

```
RecordRef[] rr = new RecordRef[]{new RecordRef("1", RecordType.customer),
new RecordRef("2", RecordType.customer), new RecordRef("3",
RecordType.customer)};
CustomerSearchBasic customerSearchBasic = new CustomerSearchBasic();
customerSearchBasic.setInternalId(new SearchMultiSelectField(rr,
SearchMultiSelectFieldOperator.anyOf));
ContactSearch contactSearch = new ContactSearch();
contactSearch.setCustomerJoin(customerSearchBasic);
```

SOAP Request

```
<soapenv:Body>
  <search xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
    <searchRecord xsi:type="ns1:ContactSearch"
      xmlns:ns1="urn:relationships_2_6.lists.webservices.netsuite.com">
      <ns1:customerJoin xsi:type="ns2:CustomerSearchBasic"
        xmlns:ns2="urn:common_2_6.platform.webservices.netsuite.com">
        <ns2:internalId operator="anyOf" xsi:type="ns3:SearchMultiSelectField"
          xmlns:ns3="urn:core_2_6.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>
```

```

        </nsl:customerJoin>
    </searchRecord>
</search>
</soapenv:Body>

```

Search Preferences

The SearchPreferences type is used to set preferences for the search. This must be set in the SOAP message header. The SearchPreference type contains the following fields.



Note: For more information and code samples of how to set preferences at the request level, refer to “Setting Preferences at the Request Level” on page 25.

Element Name	XSD Type	Notes
bodyFieldsOnly	boolean	Defaults to TRUE and indicates that the information in the body fields of the record are returned — significantly improving performance. Any fields in associated lists or sublists are not returned. If the bodyFieldsOnly field is set to FALSE, all fields associated with the record are returned.
pageSize	int	See Pagination below.



Important: When performing searches on records, by default only the body fields are returned. 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" />

```

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/ids, and a full search on all messages.

Pagination

Pagination is the concept of breaking up a large number of records that are part of a search result set into smaller chunks called pages. You then have the ability to retrieve these pages of records one at a time.

Pagination is mainly used to make processing such large result sets in a more manageable manner.

There are two different ways for a page size to be determined:

- **System defined maximum page size:** This is set to 1000 records for synchronous Web service requests by default
- **User defined page size parameter:** The pageSize field in the searchPreference type is used to specify a value for the page size. The value must be greater than 10 and less than the system defined maximum of 1000.

If the number of records for the search results exceeds the page size, the remaining results must be retrieved in a subsequent operation using the searchMore operation. See [page 118](#) for more information on searchMore.

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 in essence means “Any Of” all list options. Such a search would result in all records which do NOT have this list type field as null. In order to accomplish this “None Of” “-None-“ search you need to set the internalId of the search key to “@None@”.

Example:

To search for Customers which have a Partner associated with them in NetSuite, the SOAP would look as below for the “Partner field not null” part:

```
<ns3:partner operator="noneOf">
<ns8:searchValue internalId="@NONE@"
xmlns:ns8="urn:core_2_6.platform.webservices.netsuite.com"/>
</ns3:partner>
<ns3:customFieldList>
```

Java code to generate this SOAP would be:

```
// Adding search criteria where Partner is not null
RecordRef[] noPartner = new RecordRef[1];
noPartner[0] = new RecordRef();
noPartner[0].setInternalId("@NONE@");
SearchMultiSelectField partner = new SearchMultiSelectField();
partner.setOperator(SearchMultiSelectFieldOperator.noneOf);
partner.setSearchValue(noPartner);
custSearch.setPartner(partner);
```

.Net code to generate this SOAP would be:

```
// Adding search criteria where Partner is not null
RecordRef[] noPartner = new RecordRef[1];
noPartner[0] = new RecordRef();
noPartner[0].internalId = "@NONE@";
SearchMultiSelectField partner = new SearchMultiSelectField();
partner.@operator = SearchMultiSelectFieldOperator.noneOf;
partner.searchValue = noPartner;
custSearch.partner = partner;
```

Request

The SearchRequest type is used for the request. It contains the following field.

Element Name	XSD Type	Notes
searchRecord	SearchRecord	The SearchRecord type is an abstract type. An instance of a type that extends SearchRecord must be used—such as CustomerSearchBasic or EventSearchBasic.

Response

The SearchResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
status	Status	The status for this search. All applicable errors or warnings will be listed within this type.
totalRecords	xsd:int	The total number of records for this search. Depending on the pageSize value, some or all the records may be returned in this response
pageSize	xsd:int	The page size for this search.
totalPages	xsd:int	The total number of pages that are part of this search.
pageIndex	xsd:int	The page index for the current set of results.
recordList	Record[]	A list of records that meet the criteria for this search. The actual records returned need to be of a type that extends the abstract type of record.

Faults

This operation can throw one of the following faults. See “[Soap Fault Status Codes](#)” on [page 152](#) for more information on faults.

- InvalidSessionFault
- ExceededRateLimitFault
- ExceededMaxRecordsFault
- UnexpectedErrorFault

Sample Code

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 “[Pagination](#)” on [page 111](#)).

```
<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>
        </customerJoin>
      </searchRecord>
    </search>
  </soap:Body>
```

SOAP Response

Notice that in this example, only one matching record was found. You can see that the page size was set such that if multiple records were found, only 10 would be returned at a time. The [searchMore](#) or [searchNext](#) operation could then be performed to return additional results.

```
<soapenv:Body>
<searchResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
<searchResult xmlns="urn:core_2_6.platform.webservices.netsuite.com">
<status isSuccess="true"/>
<totalRecords>1</totalRecords>
<pageSize>10</pageSize>
<totalPages>1</totalPages>
<pageIndex>1</pageIndex>
<recordList>
<record internalId="983" xsi:type="nsl:Customer"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:nsl="urn:relationships_2_6.lists.webservices.netsuite.com">
<nsl:entityId>Shutter Fly</nsl:entityId>
<nsl:isInactive>>false</nsl:isInactive>
<nsl:companyName>Shutter Fly, Inc</nsl:companyName>
<nsl:entityStatus internalId="6"><name>LEAD-New</name>
.
.
.
<nsl:customFieldList>
<customField internalId="custentity_map" xsi:type="StringCustomFieldRef">
<value>http://maps.google.com</value>
</customField>
<customField internalId="custentity_had_order_problems"
xsi:type="BooleanCustomFieldRef"><value>>false</value></customField>
</nsl:customFieldList>
</record>
</recordList>
</searchResult>
</searchResponse>
</soapenv:Body>
```

C#

```
private void searchCustomer()
{
    // This operation requires a valid session
    this.login( true );

    _out.WriteLine( "\nEnter search parameters" );

    // Instantiate a search object for customers. Note that the search
    // object is different from the regular record used for add and update.
    CustomerSearch custSearch = new CustomerSearch();

    // Search the customer entity id which is a string field
    _out.write( "Entity ID (press enter to skip): " );
    String nameValue = _out.readLn();
    SearchStringField entityId = null;
    if ( !nameValue.Equals( "" ) )
    {
        entityId = new SearchStringField();
        entityId.@operator = SearchStringFieldOperator.contains;
        entityId.operatorSpecified = true;
        entityId.searchValue = nameValue;
        custSearch.entityId = entityId;
    }
}
```

```

// Search the customer stage which is a list field
_out.write( "Customer Stage (one or more nsKeys separated by commas, press enter
to skip): " );
String stageKeysValue = _out.readLn();
SearchMultiSelectField stage = null;
if ( !stageKeysValue.Equals( "" ) )
{
    stage = new SearchMultiSelectField();
    stage.@operator = SearchMultiSelectFieldOperator.anyOf;
    stage.operatorSpecified = true;

    string [] nskeys = stageKeysValue.Split( new Char[] {','} );

    RecordRef[] recordRefs = new RecordRef[ stageKeysValue.Length ];
    for (int i=0; i<nskeys.Length; i++ )
    {
        RecordRef recordRef = new RecordRef();
        recordRef.internalId = nskeys[i];
        recordRefs[i] = recordRef;
    }
    stage.searchValue = recordRefs;
    custSearch.stage = stage;
}

// Search by isActive field which is a boolean
/*
SearchBooleanField isActive = new SearchBooleanField();
while ( true )
{
    _out.write( "Is active [T/F] (default is T): " );
    String upcomingStr = _out.readLn();

    if ( "T".Equals( upcomingStr.ToUpper() ) || "".Equals( upcomingStr.ToUpper()
) )
    {
        isActive.searchValue = true;
        isActive.searchValueSpecified = true;
        break;
    }
    else if ( upcomingStr.ToUpper().Equals( "F" ) )
    {
        isActive.searchValue = false;
        isActive.searchValueSpecified = true;
        break;
    }
    else
    {
        _out.writeLn( "Invalid selection" );
    }
}
custSearch.active = isActive;
*/
if ( custSearch.entityId == null && custSearch.stage == null )
{
    _out.info( "\nNo search criteria was specified. Searching for all records." );
}
else
{
    _out.info(
        "\nSearching for customers with the following criteria: " +
        (entityId==null ? "" : ("\nentityID=" + custSearch.entityId.searchValue) + ",
Operator=" + entityId.@operator.ToString()) +
        (stage==null ? "" : ("\nstage nsKeys='" + stageKeysValue + "',
Operator=" + stage.@operator.ToString())) );
}

```

```

}

// Invoke search() web services operation
SearchResult response = _service.search( custSearch );

// Process response
if ( response.status.isSuccess )
{
    // Process the records returned in the response and print to console
    processCustomerSearchResponse( response );

    // Since pagination controls what is returned, check to see
    // if there are anymore pages to retrieve.
    searchMore( response );
}
else
{
    _out.error( getStatusDetails( response.status ) );
}
}

```

Java

```

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

    _console.writeln("\nEnter search parameters");

    // Instantiate a search object for customers. Note that the search
    // object is different from the regular record used for add and update.
    CustomerSearch custSearch = new CustomerSearch();

    // Search the customer entity id which is a string field
    _console.write("Entity ID (press enter to skip): ");
    String nameValue = _console.readLine();
    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("\nNo search criteria was specified. Searching for all records.");
    } else {
        _console
        .info("\nSearching for customers with the following criteria: "
            + (entityId == null ? ""
                : ("\nentityId=" + custSearch
                    .getEntityId().getSearchValue()
                    + ", Operator="
                    + entityId.getOperator().toString())
                + (stage == null ? "" : ("\nstage nsKeys="
                    + stageKeysValue + ", Operator=" + stage
                    .getOperator().toString())));
    }

    // Set page size for number of records to be returned in search
    // response

    // Invoke search() web services operation
    SearchResult result = _port.search(custSearch);

    // Process result
    if (result.getStatus().isIsSuccess()) {
        // Process the records returned in the result and print to console
        processCustomerSearchResponse(result);

        // Since pagination controls what is returned, check to see
        // if there are anymore pages to retrieve.
        searchMore(result);
    } else {
        _console.error(getStatusDetails(result.getStatus()));
    }
}

```

Searching for a Multi-Select Custom Field

In the following code sample, the results for the custom transaction field 'custcolcolumnname' are returned.

```

// transaction search by custom column field
TransactionSearch transactionSearch = new TransactionSearch();

SearchCustomFieldList searchCustomFieldList = new SearchCustomFieldList();
transactionSearch.setCustomFieldList(searchCustomFieldList);

// make the multiselectsearch
SearchMultiSelectCustomField searchMultiSelectCustomField = new
SearchMultiSelectCustomField();

ListOrRecordRef listOrRecordRef = new ListOrRecordRef();
listOrRecordRef.setInternalId("1");// the internal id of the custom list entry
to match
listOrRecordRef.setName("yourcustomlistname");// the name of the list

searchCustomFieldList.setCustomField(new
SearchCustomField[]{searchMultiSelectCustomField});

// make the search expression
searchMultiSelectCustomField.setInternalId("custcolcolumnname");//the name of
the tx custom column

```

```

searchMultiSelectCustomField.setOperator(SearchMultiSelectFieldOperator.anyOf);
searchMultiSelectCustomField.setSearchValue(new ListOrRecordRef[]
{listOrRecordRef});

SearchResult sr = _port.search(transactionSearch);

```

searchMore

The searchMore operation is used to retrieve more records after an initial search operation. The results returned in a searchMore operation reflect the records in the next segment as defined in the **original** search operation. Therefore, if a record is deleted before all records have been returned, the total number of records may differ from the total record count returned in the search operation — the deleted record is not returned. If a record is **added**, that record is not returned in subsequent searchMore operations. However, the data in each record returned is current such that if a change to a record occurs in between the time of the original search operation and the searchMore operation, the updated data is returned.

Example

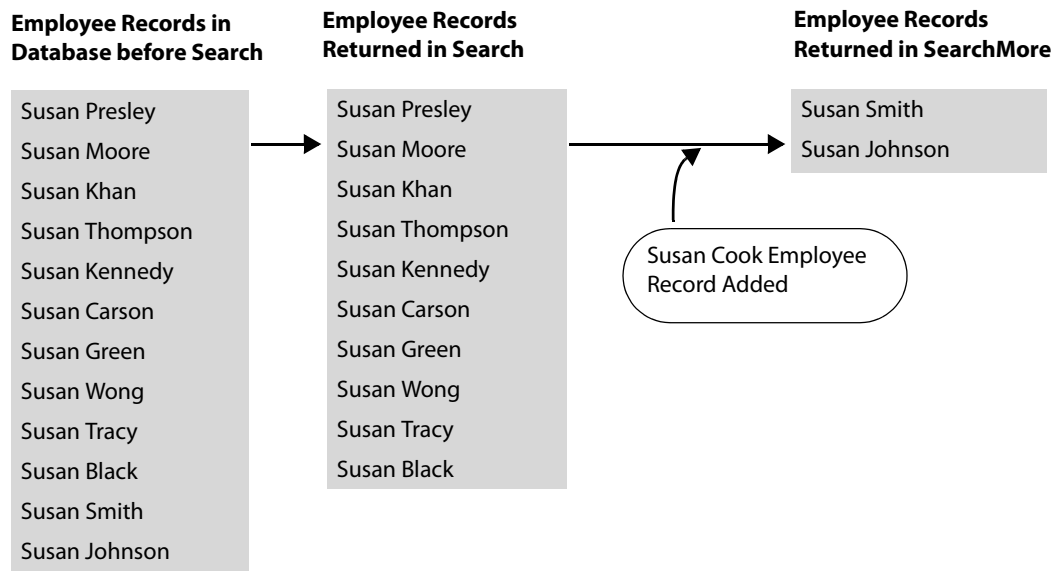
Suppose you submit the following request for all employees whose firstName contains 'Susan' and set the pageSize to 10 records:

```

<soap:Header>
  <platformMsgs:searchPreferences>
    <platformMsgs:bodyFieldsOnly>true</platformMsgs:bodyFieldsOnly>
    <platformMsgs:pageSize>10</platformMsgs:pageSize>
  </platformMsgs:searchPreferences>
</soap:Header>
<soap:Body>
  <platformMsgs:search>
    <platformMsgs:searchRecord xsi:type="listEmp:EmployeeSearch">
      <listEmp:firstName operator="contains"
xsi:type="platformCore:SearchStringField">
        <platformCore:searchValue>Susan</platformCore:searchValue>
      </listEmp:firstName>
    </platformMsgs:searchRecord>
  </platformMsgs:search>
</soap:Body>

```

As illustrated in the following figure, if the database contains 12 matches at the time of the initial Search, the Search result would provide the first 10 records as defined in the pageSize preference. A subsequent SearchMore operation would return only the remaining two records from the original search result even if another employee record, Susan Cook, is added prior to the SearchMore request being submitted.



Request

The SearchMoreRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
pageIndex	SearchRecord	An index that specifies which page in the search to return. If it is not provided, the next page is returned.

Response

The SearchMoreResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
status	Status	The status for this search. All applicable errors or warnings will be listed within this type.
searchPreferences	searchPreferences	Sets the Preferences for this search. The bodyFieldsOnly field of the searchPreference type defaults to true and indicates that the information in the body fields of the record are returned — significantly improving performance. Any fields in associated lists or sublists are not returned. If the bodyFieldsOnly field is set to false, all fields associated with the record are returned.
totalRecords	xsd:int	The total number of records for this search. Depending on the pageSize value, some or all the records may be returned in this response
pageSize	xsd:int	The page size for this search.
totalPages	xsd:int	The total number of pages that are part of this search.

Element Name	XSD Type	Notes
pageIndex	xsd:int	The page index for the current set of results.
recordList	Record[]	A list of records that meet the criteria for this search. The actual records returned need to be of a type that extends the abstract type of record.

Faults

This operation can throw one of the following faults. See “Soap Fault Status Codes” on page 152 for more information on faults.

- InvalidSessionFault
- ExceededRateLimitFault
- ExceededMaxRecordsFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In this example, page 2 of the search result set is requested.

```
<soap:Body>
  <platformMsgs:searchMore>
    <platformMsgs:pageIndex>2</platformMsgs:pageIndex>
  </platformMsgs:searchMore>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
  <searchMoreResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
    <searchResult xmlns="urn:core_2_6.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_2_6.lists.webservices.netsuite.com">
          <ns1:entityId>Jackson Alexander</ns1:entityId>
          <ns1:isInactive>>false</ns1:isInactive>
          .
          ...[more fields]
          .
          <ns1:customFieldList>
            <customField internalId="custentity_map" xsi:type="StringCustomFieldRef">
              <value>http://maps.google.com</value>
            </customField>
          </ns1:customFieldList>
        </record>
        <record internalId="227" xsi:type="ns2:Customer"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:ns2="urn:relationships_2_6.lists.webservices.netsuite.com">
          <ns2:entityId>Seena Thomas</ns2:entityId>
          <ns2:isInactive>>false</ns2:isInactive>
```

```

<ns2:companyName>Seena Thom Inc.</ns2:companyName>
.
...[more fields]
.
<ns2:customFieldList>
  <customField internalId="custentity_map" xsi:type="StringCustomFieldRef">
    <value>http://maps.google.com</value>
  </customField>
</ns2:customFieldList>
</record>
.
...[more records]
.
</recordList>
</searchResult>
</searchMoreResponse>
</soapenv:Body>

```

C#

```

private void searchMore( SearchResult response )
{
    //SearchResponse response;
    bool isGetAllPages = false;

    // Keep getting pages until there are no more pages to get
    while ( response.totalRecords > (response.pageSize * response.pageIndex) )
    {
        if ( !isGetAllPages )
        {
            // Only continue if user wants to get the next page
            _out.write( "\nThere are more search results. Would you like to get the next
page for
this search? (A/Y/N): " );
            String userResponse = _out.readLine().ToUpper();
            if ( String.Equals( userResponse, "N" ) )
            {
                break;
            }
            else if ( String.Equals( userResponse, "A" ) )
            {
                isGetAllPages = true;
            }
        }
    }

    // Invoke searchMore() operation
    response = _service.searchMore( response.pageIndex + 1 );

    // Process response
    if ( response.status.isSuccess )
    {
        processCustomerSearchResponse( response );
    }
    else
    {
        _out.error( getStatusDetails( response.status ) );
    }
}
}

```

Java

```

public void searchMore(SearchResult result) throws RemoteException,
ExceededUsageLimitFault, UnexpectedErrorFault, InvalidSessionFault,
ExceededRecordCountFault {
    // SearchResult response;

```

```

boolean isGetAllPages = false;

// Keep getting pages until there are no more pages to get
while (result.getPageSize() != null
&& result.getPageIndex() != null
&& (result.getTotalRecords().intValue() > (result.getPageSize()
.intValue() * result.getPageIndex().intValue()))) {
    if (!isGetAllPages) {
        // Only continue if user wants to get the next page
        _console
        .write("\nThere are more search results. Would you like to get the next page
for
        this search? (A/Y/N):");
        String userResponse = _console.readLine().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()));
    }
}
}

```

searchNext

The searchNext operation is used to retrieve the next set of records after an initial search operation.

Request

The searchNextRequest type is used for the request. It does not contain any fields.

Response

The searchMoreResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
status	Status	The status for this search. All applicable errors or warnings will be listed within this type.
totalRecords	xsd:int	The total number of records for this search. Depending on the pageSize value, some or all the records may be returned in this response
pageSize	xsd:int	The page size for this search.
totalPages	xsd:int	The total number of pages that are part of this search.

Element Name	XSD Type	Notes
pageIndex	xsd:int	The page index for the current set of results.
recordList	Record[]	A list of records that meet the criteria for this search. The actual records returned need to be of a type that extends the abstract type of record.

Faults

This operation can throw one of the following faults. See “Soap Fault Status Codes” on page 152 for more information on faults.

- InvalidSessionFault
- ExceededRateLimitFault
- ExceededMaxRecordsFault
- UnexpectedErrorFault

Sample Code

SOAP Request

```
<soap:Body>
  <platformMsgs:searchNext/>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
<searchNextResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
<searchResult xmlns="urn:core_2_6.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="nsl:Customer"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:nsl="urn:relationships_2_6.lists.webservices.netsuite.com">
    <nsl:entityId>John Bauer</nsl:entityId>
    <nsl:isInactive>>false</nsl:isInactive>
    .
    ...[more fields]
    .
    <nsl:customFieldList>
      <customField internalId="custentity_map" xsi:type="StringCustomFieldRef">
        <value>http://maps.google.com</value>
      </customField>
    </nsl:customFieldList>
  </record>
  .
  ...[more records]
  .
</recordList>
</searchResult>
</searchNextResponse>
</soapenv:Body>
```

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.

Request

The `getListRequest` type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
recordRef	RecordRef	An array of recordRef objects that specify the ids of the records to be retrieved.

Response

The `getListResponse` type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
recordList	Record[]	A list of records that correspond to the specified ids. The actual records returned need to be of a type that extends the abstract type Record.

Faults

This operation can throw one of the following faults. See “[Soap Fault Status Codes](#)” on [page 152](#) for more information on faults.

- InvalidSessionFault
- ExceededRateLimitFault
- ExceededMaxRecordsFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In the following example, 2 records are retrieved — one customer record and one employee record. Note that you must provide the internal ID of the specify instance of the record and the record type for the `getList`.

```

<soap:Body>
<platformMsgs:getList>
  <platformMsgs:baseRef internalId="983" type="customer"
xsi:type="platformCore:RecordRef"/>
  <platformMsgs:baseRef internalId="-5" type="employee"
xsi:type="platformCore:RecordRef"/>
</platformMsgs:getList>
</soap:Body>

```

SOAP Response

```

<soapenv:Body>
<getListResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
<readResponseList xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
  <readResponse>
    <ns1:status isSuccess="true"
xmlns:ns1="urn:core_2_6.platform.webservices.netsuite.com"/>
    <record internalId="983" xsi:type="ns2:Customer"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns2="urn:relationships_2_6.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="custentity_map"
xsi:type="ns6:StringCustomFieldRef"
xmlns:ns6="urn:core_2_6.platform.webservices.netsuite.com">
          <ns6:value>http://maps.google.com</ns6:value>
        </ns6:customField>
      </ns2:customFieldList>
    </record>
  </readResponse>
</readResponseList>
</getListResponse>
</soapenv:Body>

```

C#

```

private int getCustomerList()
{
  // This operation requires a valid session
  this.login( true );

  // Prompt for list of nsKeys and put in an array
  _out.write( "\nnsKeys for records to retrieved (separated by commas): " );
  String reqKeys = _out.readLine();
  string [] nsKeys = reqKeys.Split( new Char[] {','} );

  return getCustomerList( nsKeys, false );
}

```

Java

```

public int getCustomerList() throws RemoteException,
ExceededUsageLimitFault, UnexpectedErrorFault, InvalidSessionFault,
ExceededRecordCountFault {
    // This operation requires a valid session
    this.login(true);

    // Prompt for list of nsKeys and put in an array
    _console
.write("\nnsKeys for records to retrieved (separated by commas): ");
String reqKeys = _console.readLine();
String[] nsKeys = reqKeys.split(",");

    return getCustomerList(nsKeys, false);
}

```

get

The get operation is used to retrieve a record by providing the unique id that identifies that record.

Request

The getRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
recordRef	RecordRef	A recordRef object that specifies the id of the record to be retrieved.

Response

The getListResponse type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
record	Record	A record that represents the specified id. The actual record returned needs to be a type that extends the abstract type Record.

Faults

This operation can throw one of the following faults. See “Soap Fault Status Codes” on [page 152](#) for more information on faults.

- InvalidSessionFault
- ExceededRateLimitFault
- ExceededMaxRecordsFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In this example, a single customer record is retrieved. Note that the internal ID for the specific instance of the record and the record type (customer) must be specified.

```
<soap:Body>
  <platformMsgs:get>
    <platformMsgs:baseRef internalId="983" type="customer"
    xsi:type="platformCore:RecordRef">
      <platformCore:name/>
    </platformMsgs:baseRef>
  </platformMsgs:get>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
  <getResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
  <readResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
    <ns1:status isSuccess="true"
    xmlns:ns1="urn:core_2_6.platform.webservices.netsuite.com"/>
    <record internalId="983" xsi:type="ns2:Customer"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:ns2="urn:relationships_2_6.lists.webservices.netsuite.com">
      <ns2:entityId>Shutter Fly</ns2:entityId><ns2:isInactive>>false</
    ns2:isInactive>
      <ns2:companyName>Shutter Fly, Inc</ns2:companyName>
      <ns2:entityStatus internalId="6">
        .
        ...[more fields]
        .
      <ns2:customFieldList>
        <ns6:customField internalId="custentity_map"
        xsi:type="ns6:StringCustomFieldRef"
        xmlns:ns6="urn:core_2_6.platform.webservices.netsuite.com">
          <ns6:value>http://maps.google.com</ns6:value>
        </ns6:customField>
      </ns2:customFieldList>
    </record>
  </readResponse>
</getResponse>
</soapenv:Body>
```

C#

```
private void getCustomer()
{
  // This operation requires a valid session
  this.login( true );

  // Prompt for the nsKey
  _out.write( "\nnsKey for record to be retrieved: " );
  String nsKey = _out.readLine();

  // Invoke the get() operation to retrieve the record
  RecordRef recordRef = new RecordRef();
  recordRef.internalId = nsKey;
  recordRef.type = RecordType.customer;
  recordRef.typeSpecified = true;

  ReadResponse response = _service.get( recordRef );

  // Process response from get() operation
```

```

_out.info( "\nRecord returned from get() operation: " );
if ( !response.status.isSuccess )
{
    _out.info(
        "ERROR: " +
        getStatusDetails( response.status ) );
}
else
{
    Customer customer = (Customer) response.record;
    _out.info(
        "\nnsKey=" + customer.internalId + ", " +
        "\nentityId=" + customer.entityId +
        (customer.companyName==null ? "" : ("\ncompanyName=" + customer.companyName))
+
        (customer.stage==null ? "" : ("\nstage=" + customer.stage)) +
        (customer.email==null ? "" : ("\nemail=" + customer.email)) +
        (customer.phone==null ? "" : ("\nphone=" + customer.phone)) +
        "\nisInactive=" + customer.isInactive +
        (!customer.dateCreatedSpecified ? "" : ("\ndateCreated=" +
        customer.dateCreated.ToShortDateString())) );
}
}
}

```

Java

```

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

    // Prompt for the nsKey
    _console.write("\nnsKey for record to be retrieved: ");
    String nsKey = _console.readLine();

    // Invoke the get() operation to retrieve the record
    RecordRef recordRef = new RecordRef();
    recordRef.setInternalId(nsKey);
    recordRef.setType(RecordType.customer);

    ReadResponse response = _port.get(recordRef);

    // Process response from get() operation
    _console.info("\nRecord returned from get() operation: ");
    if (!response.getStatus().isSuccess()) {
        _console.info("ERROR: " + getStatusDetails(response.getStatus()));
    } else {
        Customer customer = (Customer) response.getRecord();
        _console
        .info("\nnsKey="
        + customer.getInternalId()
        + ", "
        + "\nentityId="
        + customer.getEntityId()
        + (customer.getCompanyName() == null ? ""
        : ("\ncompanyName=" + customer
        .getCompanyName()))
        + (customer.getStage() == null ? ""
        : ("\nstage=" + customer.getStage()))
        + (customer.getEmail() == null ? ""
        : ("\nemail=" + customer.getEmail()))
        + (customer.getPhone() == null ? ""
        : ("\nphone=" + customer.getPhone()))
        + "\nisInactive="
        + customer.getIsInactive()

```

```

    + (customer.getDateCreated() != null ? ""
      : ("\ndateCreated=" + customer
        .getDateCreated().toString()));
  }
}

```

getAll

The getAll operation is used to retrieve a list of all records of the specified type. Records that support the getAll operation are listed in the GetAllRecordType as defined in the platformCoreType system constants XSD file:

```

<!-- getAll Record Types -->
<simpleType name="GetAllRecordType">
  <restriction base="xsd:string">
    <enumeration value="customerCategory" />
    <enumeration value="contactCategory" />
    <enumeration value="priceLevel" />
    <enumeration value="winLossReason" />
    <enumeration value="term" />
    <enumeration value="noteType" />
    <enumeration value="paymentMethod" />
    <enumeration value="leadSource" />
  </restriction>
</simpleType>

```



Note: These are all records of the **Other List** type.

Request

The getAllRequest type is used for the request. It contains the following fields.

Element Name	XSD Type	Notes
recordType	GetAllRecordType	Specify the record type.

Response

The getList response type is used for the response. It contains the following fields.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
recordList	Record[]	A list of records that correspond to the specified ids. The actual records returned need to be of a type that extends the abstract type Record.

Faults

This operation can throw one of the following faults. See “Soap Fault Status Codes” on page 152 for more information on faults.

- InvalidSessionFault
- ExceededRateLimitFault

- ExceededMaxRecordsFault
- UnexpectedErrorFault

Sample Code

SOAP Request

In this example, all customer category records for the currently logged in account are retrieved.

```
<soap:Body>
<platformMsgs:getAll>
  <platformMsgs:record recordType="customerCategory"/>
</platformMsgs:getAll>
</soap:Body>
```

SOAP Response

```
<soapenv:Body>
<getAllResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
<getAllResult xmlns="urn:core_2_6.platform.webservices.netsuite.com">
<status isSuccess="true"/>
<totalRecords>2</totalRecords>
<recordList>
  <record internalId="1" xsi:type="ns1:CustomerCategory"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns1="urn:accounting_2_6.lists.webservices.netsuite.com">
  <ns1:name>Corporate</ns1:name>
  <ns1:inactive>>false</ns1:inactive>
</record>
  <record internalId="2" xsi:type="ns2:CustomerCategory"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ns2="urn:accounting_2_6.lists.webservices.netsuite.com">
  <ns2:name>Individual</ns2:name>
  <ns2:inactive>>false</ns2:inactive>
</record>
</recordList>
</getAllResult>
</getAllResponse>
</soapenv:Body>
```

C#

```
private void getAll()
{
  // This operation requires a valid session
  this.login( true );

  // Instantiate GetAllRecord and set the record type
  GetAllRecord record = new GetAllRecord();
  record.recordTypeSpecified = true;

  _out.WriteLine( "\nSelect the list type to be retrieved");
  _out.WriteLine( "1) Customer Categories");
  _out.WriteLine( "2) Contact Categories" );
  _out.WriteLine( "3) Price Levels" );
  _out.WriteLine( "4) Win Loss Reasons" );
  _out.WriteLine( "5) Terms" );
  _out.WriteLine( "6) Note Types" );
  _out.WriteLine( "7) Payment Methods" );
  _out.WriteLine( "8) Lead Sources" );
  _out.write( "\nSelection: " );
  String strMyChoice = _out.ReadLine().ToUpper();
  _out.write( "\n" );
}
```

```
int choice = Convert.ToInt32( strMyChoice );

switch ( choice )
{
    case 1:
        record.recordType = GetAllRecordType.customerCategory;
        break;
    case 2:
        record.recordType = GetAllRecordType.contactCategory;
        break;
    case 3:
        record.recordType = GetAllRecordType.priceLevel;
        break;
    case 4:
        record.recordType = GetAllRecordType.winLossReason;
        break;
    case 5:
        record.recordType = GetAllRecordType.term;
        break;
    case 6:
        record.recordType = GetAllRecordType.noteType;
        break;
    case 7:
        record.recordType = GetAllRecordType.paymentMethod;
        break;
    case 8:
        record.recordType = GetAllRecordType.leadSource;
        break;
}

// Invoke getAll() operation
GetAllResult result = _service.getAll( record );
Record[] records = result.recordList;

if ( result.status.isSuccess )
{
    _out.info(
        "\nThe requested list for " + record.recordType.ToString() +
        " returned " + result.totalRecords + " records:" );
    int numRecords = 0;
    for (int i=0; i<records.Length; i++ )
    {
        numRecords++;
        switch ( choice )
        {
            case 1:
                CustomerCategory custCategory = (CustomerCategory) records[i];
                _out.info( "key=" + custCategory.internalId + ", name=" +
                    custCategory.name );
                break;
            case 2:
                ContactCategory contCategory = (ContactCategory) records[i];
                _out.info( "key=" + contCategory.internalId + ", name=" +
                    contCategory.name );
                break;
            case 3:
                PriceLevel priceLevel = (PriceLevel) records[i];
                _out.info( "key=" + priceLevel.internalId + ", name=" + priceLevel.name
                );
                break;
            case 4:
                WinLossReason winLossReason = (WinLossReason) records[i];
                _out.info( "key=" + winLossReason.internalId + ", name=" +
                    winLossReason.name );
        }
    }
}
```

```

        break;
    case 5:
        Term term = (Term) records[i];
        _out.info( "key=" + term.internalId + ", name=" + term.name );
        break;
    case 6:
        NoteType noteType = (NoteType) records[i];
        _out.info( "key=" + noteType.internalId + ", name=" + noteType.name );
        break;
    case 7:
        PaymentMethod paymentMethod = (PaymentMethod) records[i];
        _out.info( "key=" + paymentMethod.internalId + ", name=" +
            paymentMethod.name );
        break;
    case 8:
        LeadSource leadSource = (LeadSource) records[i];
        _out.info( "key=" + leadSource.internalId + ", name=" + leadSource.name
    );
        break;
    }
} // for
}
}

```

Java

```

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

    // Instantiate GetAllRecord and set the record type
    GetAllRecord record = new GetAllRecord();

    _console.writeln("\nSelect the list type to be retrieved");
    _console.writeln("1) Customer Categories");
    _console.writeln("2) Contact Categories");
    _console.writeln("3) Price Levels");
    _console.writeln("4) Win Loss Reasons");
    _console.writeln("5) Terms");
    _console.writeln("6) Note Types");
    _console.writeln("7) Payment Methods");
    _console.writeln("8) Lead Sources");
    _console.write("\nSelection: ");
    String strMyChoice = _console.readLine().toUpperCase();
    _console.write("\n");

    int choice = Integer.parseInt(strMyChoice);

    switch (choice) {
        case 1:
            record.setRecordType(GetAllRecordType.customerCategory);
            break;
        case 2:
            record.setRecordType(GetAllRecordType.contactCategory);
            break;
        case 3:
            record.setRecordType(GetAllRecordType.priceLevel);
            break;
        case 4:
            record.setRecordType(GetAllRecordType.winLossReason);
            break;
        case 5:
            record.setRecordType(GetAllRecordType.term);
            break;
    }
}

```

```
case 6:
    record.setRecordType(GetAllRecordType.noteType);
    break;
case 7:
    record.setRecordType(GetAllRecordType.paymentMethod);
    break;
case 8:
    record.setRecordType(GetAllRecordType.leadSource);
    break;
}

// Invoke getAll() operation
GetAllResult result = _port.getAll(record);

// Process results
RecordList recordList = result.getRecordList();
Record[] records = recordList.getRecord();
if (result.getStatus().isIsSuccess()) {
    _console.info("\nThe requested list for "
        + record.getRecordType().toString() + " returned "
        + result.getTotalRecords() + " records:");
    int numRecords = 0;
    for (int i = 0; i < records.length; i++) {
        numRecords++;
        switch (choice) {
            case 1:
                CustomerCategory custCategory = (CustomerCategory) records[i];
                _console.info("key=" + custCategory.getInternalId() + ", name="
                    + custCategory.getName());
                break;
            case 2:
                ContactCategory contCategory = (ContactCategory) records[i];
                _console.info("key=" + contCategory.getInternalId() + ", name="
                    + contCategory.getName());
                break;
            case 3:
                PriceLevel priceLevel = (PriceLevel) records[i];
                _console.info("key=" + priceLevel.getInternalId() + ", name="
                    + priceLevel.getName());
                break;
            case 4:
                WinLossReason winLossReason = (WinLossReason) records[i];
                _console.info("key=" + winLossReason.getInternalId() + ", name="
                    + winLossReason.getName());
                break;
            case 5:
                Term term = (Term) records[i];
                _console.info("key=" + term.getInternalId() + ", name="
                    + term.getName());
                break;
            case 6:
                NoteType noteType = (NoteType) records[i];
                _console.info("key=" + noteType.getInternalId() + ", name="
                    + noteType.getName());
                break;
            case 7:
                PaymentMethod paymentMethod = (PaymentMethod) records[i];
                _console.info("key=" + paymentMethod.getInternalId() + ", name="
                    + paymentMethod.getName());
                break;
            case 8:
                LeadSource leadSource = (LeadSource) records[i];
                _console.info("key=" + leadSource.getInternalId() + ", name="
                    + leadSource.getName());
        }
    }
}
```

```

        break;
    }
} // for
}
}

```

getSelectValue

The getSelectValue operation is used to retrieve valid values for a given recordRef field where the referenced record type is not yet exposed in the Web services API or when the logged in role does not have permission to the instances of the record type.



Important: Slaving that would normally occur in the UI is NOT performed when accessing this list via Web services. The list of values that returned reflects the state of the list as it would be on initial load of a given form. Also, the type is null if the record type is not yet exposed in the platform, or if the backend is unable to determine the type.

Request

The getSelectValueRequest type is used for the request. It contains the following field.

Element Name	XSD Type	Notes
fieldName	Enum	This enum contains a fully qualified reference to exposed multiselect fields that can be called using this operation. For example, the value that corresponds to attendees on an event record is calendarEvent.attendeeList.attendee — where calendarEvent is the record type, attendeeList is the machine on the record, and attendee is the desired multi-select field.

You can also set filter criteria for a given request to specify that a subset of choices be returned. The filter takes a string value based on the choices available in the specified drop-down list.



Important: The getSelectValue operation uses the **contains** operator in the search filter. Note that the contains operator is not case sensitive.

In the following SOAP request, all values from the employee_rolesList_selectedRole drop-down list that also contain 'Administrator' will be returned.

```

<soapenv:Body>
<getSelectValue xmlns="urn:messages_1_3.platform.webservices.netsuite.com">
  <fieldName fieldType="employee_rolesList_selectedRole">
    <ns1:searchCriteria
xmlns:ns1="urn:core_1_3.platform.webservices.netsuite.com">Administrator
  </ns1:searchCriteria>
  </fieldName>
</getSelectValue>
</soapenv:Body>

```

Response

The getSelectValueResponse type is used for the response. It contains the following fields.



Note: A maximum of 1,000 values are returned. If the actual number of values available for the requested type exceeds 1,000 a warning is returned.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
recordList	Record[]	A list of recordRefs that represent valid values available for the current field (includes the type and name).
totalRecords	int	number of choices

Faults

This operation can throw one of the following faults. See “Soap Fault Status Codes” on page 152 for more information on faults.

- InvalidSessionFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- UnexpectedErrorFault

Sample Code

SOAP Request

```
<soapenv:Body>
  <getSelectValue xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
    <fieldName fieldType="customer_customForm"/>
  </getSelectValue>
</soapenv:Body>
```

SOAP Response

```
<soapenv:Body>
  <getSelectValueResponse
xmlns="urn:messages_1_2.platform.webservices.netsuite.com">
  <getSelectValueResult xmlns="urn:core_1_2.platform.webservices.netsuite.com">
    <status isSuccess="true"/>
    <totalRecords>4</totalRecords>
    <recordRefList>
      <recordRef internalId="5">
        <name>Customer Form - Automation</name>
      </recordRef>
      <recordRef internalId="6">
        <name>Customer Form - Custom Code</name>
      </recordRef>
      <recordRef internalId="-2">
        <name>Standard Customer Form</name>
      </recordRef>
      <recordRef internalId="-5">
        <name>Standard Pop-up Customer Form</name>
      </recordRef>
    </recordRefList>
  </getSelectValueResult>
</getSelectValueResponse>
</soapenv:Body>
```

Java

```
GetSelectValueResult gr = port.getSelectValue( new GetSelectValueField
(GetSelectValueType.fromValue ("customer_customForm")));
```

Or

```
GetSelectValueResult gr = port.getSelectValue
( new GetSelectValueField (GetSelectValueType.customer_category));
```

GetSelectValue Types

For a complete list of all of the values currently supported by the getSelectValue operation, refer to the GetSelectValueType enumerations in the [coreTypes XSD](#).

getCustomization

Because any record in NetSuite can be fully customized to suit the needs of a given business, it is critical to consider these customizations when developing generic web services applications. Use the getCustomization operation to dynamically retrieve and manage the metadata for Custom Fields, Lists, and Record Types in order to handle some of the custom business logic included at the record level.

Following are the custom objects currently supported by the getCustomization operation. These are enumerated in the [coreTypes XSD](#).

- crmCustomField
- customList
- customRecordType
- entityCustomField
- itemCustomField
- itemOptionCustomField
- otherCustomField
- transactionBodyCustomField
- transactionColumnCustomField

Note: Normally, you can NOT add or update the internalId of an object. Customization is an exception to this rule. You can specify the ID on add, but can NOT update the ID thereafter. InternalIds for custom objects can be set to any unique alphanumeric string up to 30 characters long. This string must not include any spaces but can include underscores ("_").

Request

The getCustomizationRequest type is used for this request. It contains the following fields:

Element Name	XSD Type	Notes
customizationType	Customization Type	

Response

The getCustomizationResult type is used for the Response. It contains the following fields:

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
totalRecords	int	
recordList	Record[]	A list of records that correspond to the specified customization type.

Sample Code

SOAP Request

```
<soap:Body>
  <platformMsgs:getCustomization>
    <platformMsgs:customizationType getCustomizationType="crmCustomField"/>
  </platformMsgs:getCustomization>
</soap:Body>
</soap:Envelope>
```

SOAP Response

```
<soapenv:Body>
  <getCustomizationResponse
    xmlns="urn:messages_2_0.platform.webservices.netsuite.com">
    <getCustomizationResult xmlns="urn:core_2_6.platform.webservices.netsuite.com">
      <status isSuccess="true"/>
      <totalRecords>1</totalRecords>
      <recordList>
        <record internalId="CUSTEVENT1" xsi:type="ns1:CrmCustomField"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:ns1="urn:customization_2_6.setup.webservices.netsuite.com">
          <ns1:label>test</ns1:label>
          <ns1:fieldType>_freeFormText</ns1:fieldType>
          <ns1:storeValue>true</ns1:storeValue>
          <ns1:showInList>>false</ns1:showInList>
          <ns1:isParent>>false</ns1:isParent>
          <ns1:displayType>_normal</ns1:displayType>
          <ns1:isMandatory>>false</ns1:isMandatory>
          <ns1:defaultChecked>>false</ns1:defaultChecked>
          <ns1:isFormula>>false</ns1:isFormula>
          <ns1:appliesToTask>>false</ns1:appliesToTask>
          <ns1:appliesToPhoneCall>>false</ns1:appliesToPhoneCall>
          <ns1:appliesToEvent>>false</ns1:appliesToEvent>
          <ns1:appliesToCase>>false</ns1:appliesToCase>
          <ns1:appliesToCampaign>>false</ns1:appliesToCampaign>
          <ns1:appliesPerKeyword>>false</ns1:appliesPerKeyword>
          <ns1:appliesToSolution>>false</ns1:appliesToSolution>
          <ns1:availableExternally>>false</ns1:availableExternally>
        </record>
      </recordList>
    </getCustomizationResult>
  </getCustomizationResponse>
</soapenv:Body>
```

Java

```
port.getCustomization(new
CustomizationType(GetCustomizationType.transactionBodyCustomField));
```

getItemAvailability

The getItemAvailability operation can be used to retrieve the inventory availability for a given 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.

Request

The GetItemAvailabilityRequest type is used for the request.

Element Name	XSD Type	Notes
itemAvailabilityFilter	ItemAvailabilityFilter	You can filter the returned itemAvailability using this filter.

ItemAvailabilityFilter

Element Name	XSD Type	Notes
item	RecordRefList	References an existing item record in NetSuite.
lastQtyAvailableChange	dateTime	If set, only items with quantity available changes recorded as of the specified date are returned.

Response

The GetItemAvailabilityResult type is used for the response.

Element Name	XSD Type	Notes
status	Status	The status for this operation. All applicable errors or warnings are listed within this type.
itemAvailabilityList	List	Returns a list of available items.

ItemAvailabilityList

Element Name	XSD Type	Notes
item	RecordRef	References an existing item record.
lastQtyAvailableChange	dateTime	If set, only items with quantity available changes recorded as of this date are returned.
locationId	RecordRef	References a location in a user defined list at Lists > Accounting > Locations.
quantityOnHand	double	The number of units of an item in stock.
onHandValueMli	double	
reorderPoint	double	The stock level at which a new order for the item needs to be placed
preferredStockLevel	double	The preferred quantity of this item maintained in inventory for a specific location.
quantityOnOrder	double	The number of units of an item pending receipt from a vendor.

Element Name	XSD Type	Notes
quantityCommitted	double	The number of units of an item reserved by unfulfilled sales orders.
quantityBackOrdered	double	The number of units of an item reserved by unfulfilled sales orders.
quantityAvailable	double	The number of units in stock that have not been committed to fulfill sales.

ItemAvailabilityFilter

Element Name	XSD Type	Notes
item	RecordRefList	
lastQtyAvailableChange	dateTime	

Faults

This operation can throw one of the following faults. See “[Soap Fault Status Codes](#)” on [page 152](#) for more information on faults.

- InvalidSessionFault
- ExceededRequestLimitFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- UnexpectedErrorFault

Sample Code

SOAP Request

```
<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

```
<soapenv:Body>
  <getItemAvailabilityResponse
    xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
    <getItemAvailabilityResult
      xmlns="urn:core_2_6.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>

```

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.

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: Contact records can be attached to all entity records except for other Contact or Group records.

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.



Important: A user error is thrown if you attempt to attach files or records that do not exist.

The following tables lists all records that support the attach/detach operations. It also lists which records can accept file attachments as well as which records can be attached to Contact records.

Record Type	Accepts File Attachments	Accepts Contact Record Attachments
Transactions		
Check	X	
Inventory Adjustment	X	
Item Fulfillment	X	

Record Type	Accepts File Attachments	Accepts Contact Record Attachments
Journal Entry	X	
Intercompany Journal Entry	X	
Opportunity	X	
Sales Order	X	
Customer Payment	X	
Return Authorization	X	
Credit Memo	X	
Cash Refund	X	
Estimate	X	
Invoice	X	
Cash Sale	X	
Purchase Order	X	
Purchase Order Receipt	X	
Customer Payment	X	
Customer Refund	X	
Customer Deposit	X	
Customer Deposit Application	X	
Vendor Bill	X	
Entities		
Customer	X	X
Contact	X	
Employee	X	X
Partner	X	X
Vendor	X	X
Job	X	X
Group	X	
Activities		
Task	X	
Event	X	
Phone Call	X	
Support		
Case	X	
Issue	X	
Marketing		
Campaign	X	

Request (attach)

The AttachRequest type is used for this request. It contains the following fields:

Element Name	XSD Type	Notes
attachReference	AttachReference	

Request (detach)

The DetachRequest type is used for this request. It contains the following fields:

Element Name	XSD Type	Notes
attachReference	AttachReference	

Response (attach)

The AttachResponse type is used for this response. It contains the following fields:

Element Name	XSD Type	Notes
response	WriteResponse	

Response (detach)

The DetachResponse type is used for this response. It contains the following fields:

Element Name	XSD Type	Notes
response	WriteResponse	

Faults

The attach and detach operations can throw one of the following faults. See “[Soap Fault Status Codes](#)” on page 152 for more information on faults.

- UnexpectedErrorFault
- InvalidSessionFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- ExceededRequestLimitFault

Sample Code

SOAP Request (attach)

```
<attach xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
  <attachReferece xsi:type="ns1:AttachContactReference"
    xmlns:ns1="urn:core_2_6.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>
  </attachReferece>
</attach>
```

```

        </ns1:attachTo>
        <ns1:contact internalId="1467" xsi:type="ns1:RecordRef"/>
        <ns1:contactRole internalId="-10" xsi:type="ns1:RecordRef">
            <ns1:name xsi:type="xsd:string">Primary Contact</ns1:name>
        </ns1:contactRole>
    </attachReferece>
</attach>

```

SOAP Request (detach)

```

<detach xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
    <attachReferece xsi:type="ns1:AttachBasicReference"
        xmlns:ns1="urn:core_2_6.platform.webservices.netsuite.com">
        <ns1:attachTo internalId="176" type="customer" xsi:type="ns1:RecordRef">
            <ns1:name xsi:type="xsd:string">Adelina Shonkwiler</ns1:name>
        </ns1:attachTo>
        <ns1:attachedRecord internalId="1467" type="contact" xsi:type="ns1:RecordRef"/>
    </attachReferece>
</detach>

```

SOAP Response (attach)

```

<attachResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
    <writeResponse>
        <ns1:status isSuccess="true"
            xmlns:ns1="urn:core_2_6.platform.webservices.netsuite.com"/>
        <baseRef internalId="176" type="customer" xsi:type="ns2:RecordRef"
            xmlns:ns2="urn:core_2_6.platform.webservices.netsuite.com">
            <ns2:name>Adelina Shonkwiler</ns2:name>
        </baseRef>
    </writeResponse>
</attachResponse>

```

SOAP Response (detach)

```

<detachResponse xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
    <writeResponse>
        <ns1:status isSuccess="true"
            xmlns:ns1="urn:core_2_6.platform.webservices.netsuite.com"/>
        <baseRef internalId="176" type="customer" xsi:type="ns2:RecordRef"
            xmlns:ns2="urn:core_2_6.platform.webservices.netsuite.com">
            <ns2:name>Adelina Shonkwiler</ns2:name>
        </baseRef>
    </writeResponse>
</detachResponse>

```

Java (attach operation)

```

public void attach() throws Exception
{
    RecordRef contactRef = new RecordRef();
    contactRef.setInternalId("1467");
    contactRef.setType(RecordType.contact);

    RecordRef contactRoleRef = new RecordRef();
    contactRoleRef.setInternalId("-10");
    contactRoleRef.setName("Primary Contact");

    RecordRef customerRef = new RecordRef();
    customerRef.setInternalId("176");
    customerRef.setType(RecordType.customer);
}

```

```

AttachContactReference attachRef = new AttachContactReference();
attachRef.setContact(contactRef);
attachRef.setAttachTo(customerRef);
attachRef.setContactRole(contactRoleRef);

WriteResponse attachResponse = sessMgr.getPort().attach(attachRef);
}

```

Java (detach operation)

```

public void detach() throws Exception
{
    RecordRef contactRef = new RecordRef();
    contactRef.setInternalId("1467");
    contactRef.setType(RecordType.contact);

    RecordRef customerRef = new RecordRef();
    customerRef.setInternalId("176");
    customerRef.setType(RecordType.customer);

    AttachContactReference detachRef = new AttachContactReference();
    detachRef.setContact(contactRef);
    detachRef.setAttachTo(customerRef);

    WriteResponse detachResponse = sessMgr.getPort().detach(detachRef);
}

```

changePasswordOrEmail

Use the changePasswordOrEmail operation to change a user's email or password.

getDeleted

The getDeleted operation is used to retrieve a list of deleted records of a given type during a specified period. The response does not differentiate between which records were deleted via Web services versus deleted via the UI. This operation is useful in order to easily synchronize information in a client application to the data currently in NetSuite.

For example, an Outlook client application plugin 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.

Deleted records in NetSuite are tracked by internalID and record type. In cases where internalIDs are NOT unique within a record type, the deletions are NOT tracked and therefore can not be retrieved using the getDeleted operation. Examples include items, bins, and serial numbers. For a complete list of record types, refer to the DeletedRecordType enumeration in the [coreTypes xsd](#).



Important: Entity records are unique in that you may have an entity that belongs to several different subtypes. For example, a partner that is also a contact. If the partner record is deleted, the contact record remains intact.

Request

The GetDeletedRequest type is used for the request.

Element Name	XSD Type	Notes
getDeletedFilter	GetDeletedFilter	See GetDeletedFilter .

GetDeletedFilter

The GetDeletedFilter allows you to retrieve data based on the date or record type of the deleted record.

Element Name	XSD Type	Notes
deletedDate	SearchDateField	A search filter with all the available search operations. For example, search for records that were deleted since a given dateTime or between two dateTime criteria.
type	SearchEnumMultiSelectField	The type of the record that was deleted. For a complete list of all of the values currently supported by the getDeleted operation, refer to the DeletedRecordType enumerations in the coreTypes XSD .

Response

The DeletedRecord type is used for the response.

Element Name	XSD Type	Notes
deletedDate	dateTime	The status for this operation. All applicable errors or warnings are listed within this type.
record	BaseRef	Returns a list of available items.

Faults

This operation can throw one of the following faults. See “[Soap Fault Status Codes](#)” on [page 152](#) for more information on faults.

- InvalidSessionFault
- ExceededUsageLimitFault
- ExceededRecordCountFault
- UnexpectedErrorFault

Sample Code

SOAP Request

```
<soapenv:Body>
  <platformMsgs:getDeleted xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```

xmlns:platformCoreType="urn:types.core_2_5.platform.webservices.netsuite.com"
xmlns:platformCore="urn:core_2_5.platform.webservices.netsuite.com"
xmlns:platformMsgs="urn:messages_2_5.platform.webservices.netsuite.com">
  <platformMsgs:getDeletedFilter>
    <platformCore:deletedDate>
      <platformCore:predefinedSearchValue>lastBusinessWeek
    </platformCore:predefinedSearchValue>
    </platformCore:deletedDate>
    <platformCore:type>
      <platformCore:searchValue>contact</platformCore:searchValue>
    </platformCore:type>
  </platformMsgs:getDeletedFilter>
</platformMsgs:getDeleted>
</soapenv:Body>

```

initialize / initializeList

Use the initialize operation to emulate the UI workflow by prepopulating fields on transaction line items with values from a related record. Your Web services application can then modify only the values it needs to before submitting the record.

For example, in the UI clicking Bill from a Sales Order record loads an Invoice record where fields are populated with values from the Sales Order. When loading an invoice record in Web services, you can reference the related Sales Order record to initialize fields with values from that sales order.

The following table outlines all the transaction types that can be used with the initialize operation and the valid reference types they can use.

Transaction Type	Initialize Reference	Notes
Cash Refund	Cash Sale	Sets createdFrom and populates all the lines
Cash Sale	Customer	Populates Billable tabs
Cash Sale	Estimate	
Cash Sale	Opportunity	
Cash Sale	Sales Order	
Credit Memo	Customer	
Credit Memo	Invoice	
Credit Memo	Return Authorization	
Customer Payment	Customer	
Customer Payment	Invoice	
Estimate	Opportunity	
Invoice	Customer	Defaults billable time, expense, and items
Invoice	Estimate	
Invoice	Opportunity	

Transaction Type	Initialize Reference	Notes
Invoice	Sales Order	Defaults the line items as well as billable time, expense, and items
Item Fulfillment	Sales Order	Defaults the line items for fulfillment
Item Receipt	Return Authorization	
Item Receipt	Purchase Order	
Return Authorization	Cash Sale	
Return Authorization	Invoice	
Return Authorization	Sales Order	
Sales Order	Estimate	
Sales Order	Opportunity	
Vendor Bill	Purchase Order	

Ignore Read-Only Preference

In order 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 simply ignored during the Web services request. The fields still can NOT be set.

Sample Code

SOAP Request (initialize)

```
<soapenv:Body>
<platformMsgs:initialize xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs="http://
www.w3.org/2001/XMLSchema"
  xmlns:platformCoreTyp="urn:types.core_2_6.platform.webservices.netsuite.com"
  xmlns:platformCore="urn:core_2_6.platform.webservices.netsuite.com"
  xmlns:platformMsgs="urn:messages_2_6.platform.webservices.netsuite.com">
  <platformMsgs:initializeRecord>
    <platformCore:type>invoice</platformCore:type>
    <platformCore:reference internalId="1513" type="salesOrder">
      <platformCore:name>1511</platformCore:name>
    </platformCore:reference>
  </platformMsgs:initializeRecord>
</platformMsgs:initialize>
</soapenv:Body>
```

SOAP Request (initializeList)

```
<initializeList xmlns="urn:messages_2_6.platform.webservices.netsuite.com">
  <initializeRecord>
    <ns1:type
      xmlns:ns1="urn:core_2_6.platform.webservices.netsuite.com">customerPayment
    </ns1:type>
    <ns2:reference internalId="176" type="customer"
      xmlns:ns2="urn:core_2_6.platform.webservices.netsuite.com"/>
  </initializeRecord>
  <initializeRecord>
    <ns3:type xmlns:ns3="urn:core_2_6.platform.webservices.netsuite.com">
      invoice</ns3:type>
    <ns4:reference internalId="176" type="customer"
      xmlns:ns4="urn:core_2_6.platform.webservices.netsuite.com"/>
  </initializeRecord>
</initializeList>
```

Java (initializeList)

```
InitializeRef iRef1 = new InitializeRef();
iRef1.setInternalId("176");
iRef1.setType(InitializeRefType.customer);

InitializeRecord ir1 = new InitializeRecord();
ir1.setReference(iRef1);
ir1.setType(InitializeType.customerPayment);

InitializeRecord ir2 = new InitializeRecord();
ir2.setReference(iRef1);
ir2.setType(InitializeType.invoice);
```

```
sessMgr.getPort().getNetSuitePortTypePort().initializeList(new  
InitializeRecord[]{ir1, ir2});
```

Handling Errors

See the following sections for fault, error, and warning codes:

- “Soap Faults for Each Operation” on page 150
- “Soap Fault Status Codes” on page 152
- “Error Status Codes” on page 153
- “Warning Status Codes” on page 174

Soap Faults for Each Operation

The following table lists the SOAP faults that can be thrown for each supported 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

add / update / delete throw...

INVALID_VERSION
 SESSION_TIMED_OUT
 UNEXPECTED_ERROR
 USER_ERROR
 WS_CONCUR_SESSION_DISALLWD
 WS_LOG_IN_REQD

addList / updateList / deleteList throw...

INVALID_VERSION
 MAX_RCRDS_EXCEEDED
 SESSION_TIMED_OUT
 UNEXPECTED_ERROR
 USER_ERROR
 WS_CONCUR_SESSION_DISALLWD
 WS_LOG_IN_REQD

getList/getAll throw...

INVALID_VERSION
 MAX_RCRDS_EXCEEDED
 SESSION_TIMED_OUT
 UNEXPECTED_ERROR
 USER_ERROR
 WS_CONCUR_SESSION_DISALLWD
 WS_LOG_IN_REQD

get throws...

INVALID_VERSION
 SESSION_TIMED_OUT
 UNEXPECTED_ERROR

get throws...

- USER_ERROR
- WS_CONCUR_SESSION_DISALLWD
- WS_LOG_IN_REQD

search / searchNext / searchMore throw...

- INVALID_VERSION
- MAX_RCRDS_EXCEEDED
- SESSION_TIMED_OUT
- UNEXPECTED_ERROR
- USER_ERROR
- WS_CONCUR_SESSION_DISALLWD
- WS_LOG_IN_REQD

Soap Fault Status Codes

The following table lists and defines the possible SOAP fault types and corresponding codes. For a complete description of faults and how they differ from Errors and Warnings, refer to “Handling Errors” on page 56.

Fault Name	Description
InsufficientPermissionFault	This fault is thrown when the client does not have the appropriate permissions to perform an action based on the role under which they are currently logged in. If the client (user) has more than one role, they may need to login again supplying a different role with more permissions.
InvalidAccountFault	This fault is thrown when the client attempts to login with an invalid account id.
InvalidPartnerCredentials	Partner ID or password submitted with this request is invalid.
InvalidRequestIP	Originating IP is not one of the registered IPs from which this Partner request may come.
InvalidSessionFault	This fault is thrown when the client’s session has timed out or was terminated as the result of a second Web services client establishing another session. The fault message occurs on the first request attempted after the session is terminated/timed out. For more details on how NetSuite handles sessions, refer to “Session Management” on page 42.
InvalidVersionFault	This fault is thrown in the event that the request message contains an unsupported version of the schema.
ExceededRecordCountFault	This fault is thrown in the event the maximum number of records allowed for an operation has been exceeded. For more information, see “Understanding Web Services Governance” on page 11.

Fault Name	Description
ExceededRequestLimitFault	This fault is thrown if the allowed number of concurrent requests is exceeded. For more information, see "Understanding Web Services Governance" on page 11.
UnexpectedErrorFault	This fault is thrown in the event of an occurrence of an unexpected exception.
InvalidCredentialsFault	This fault is thrown in the event of an invalid username (e-mail), password and/or role supplied in a login attempt.
AsyncFault	

Error Status Codes

The following table lists possible 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.

Error Code Returned	Long Description or Message
ACCT_DISABLED	YOUR_ACCOUNT_HAS_BEEN_INACTIVATED_BY_AN_ADMINISTRATOR;
ACCT_DISABLED	ACCT_DISABLED;
ACCT_DISABLED	THIS_ACCOUNT_HAS_BEEN_DISABLED;
ACCT_DISABLED	THIS_COMPANY_HAS_BEEN_DISABLED_PLEASE_CONTACT_A_HREFMAIL TO1_SUPPORTA_TO_REENABLE_THIS_COMPANY;
ACCT_NUMS_REQD_OR_DONT_MATCH	MISSING_ACCT_OR_ACCT_NUMBERS_DONT_MATCH;
ACCT_TEMP_DISABLED	YOU_HAVE_ENTERED_AN_INVALID_PASSWORD_ON_1_CONSECUTIVE_ATTEMPTS_ACCESS_TO_YOUR_ACCOUNT_HAS_BEEN_SUSPENDED_FOR_2_MINUTES_IF_YOU_HAVE_FORGOTTEN_YOUR_PASSWORD_PLEASE_CONTACT_CUSTOMER_SUPPORT;
ACCT_TEMP_UNAVAILABLE	TEMPORARILY_UNAVAILABLE;
ACCT_TEMP_UNAVAILABLE	WS_ACCOUNT_IS_CURRENTLY_UNAVAILABLE;
ACCT_TEMP_UNAVAILABLE	WE_ARE_CURRENTLY_PERFORMING_MAINTENANCE_ON_OUR_SYSTEM_PLEASE_TRY_AGAIN_SOON;
ACCT_TEMP_UNAVAILABLE	YOUR_ACCOUNT_IS_DISABLED_FOR_1_MORE_MINUTES_DUE_TO_2_CONSECUTIVE_FAILED_LOGIN_ATTEMPTS;
ACCT_TEMP_UNAVAILABLE	YOUR_ACCOUNT_IS_NOT_YET_READY_FOR_YOU_TO_LOG_IN_PLEASE_WAIT_AND_TRY_AGAIN;
ACCT_TEMP_UNAVAILABLE	YOUR_COMPANY_DATABASE_IS_OFFLINE;
ACCT_TEMP_UNAVAILABLE	YOUR_DATA_IS_STILL_BEING_LOADED_PLEASE_TRY_AGAIN_LATER_CONTACT_A_HREFAPPICRMSUPPORTNLBUGFORMNLTYPEBUGSPF31PROFESSIONAL_SERVICESA_IF_YOU_HAVE_QUESTIONS;
ACCT_REQD	ATTEMPTING_TO_ADJUST_PROVISIONING_FOR_A_CUSTOMER_WITHOUT_AN_EXISTING_ACCOUNT;
ACTIVE_TRANS_EXIST	THERE_ARE_ACTIVE_DIRECT_DEPOSIT_TRANSACTIONS_FOR_THIS_PAYCHECK;

Error Code Returned	Long Description or Message
ADMIN_ACCESS_REQ	AT_LEAST_ONE_ACTIVE_ADMINISTRATOR_FOR_EACH_ACCOUNT_MUST_HAVE_ACCESS;
ADMIN_ACCESS_REQ	AT_LEAST_ONE_ACTIVE_ADMINISTRATOR_FOR_THIS_ACCOUNT_MUST_HAVE_ACCESS;
ADMIN_ACCESS_REQD	ONLY_ADMINISTRATORS_MAY_ENTER_A_MEMORIZED_TRANSACTION_I N_A_CLOSED_PERIOD;
ALL_DATA_DELETE_REQD	YOU_MUST_FIRST_DELETE_ALL_THE_DATA_IN_YOUR_ACCOUNT_BEFOR E_PERFORMING_THIS_ACTION;
ALL_MTRX_SUBITEMS_OPTNS_R EQD	SOME_MATRIX_SUBITEMS_EXIST_THAT_ARENT_INCLUDED_IN_THE_OPTI ONS_YOU_JUST_SPECIFIED_ON_THE_MATRIX_TAB_PLEASE_MAKE_SURE _THE_OPTIONS_YOU_SELECT_INCLUDE_ALL_EXISTING_SUBITEMS;
ALREADY_IN_INVT	THE_FOLLOWING_1_NUMBER_IS_ALREADY_IN_INVENTORY_2;
ALREADY_IN_INVT	THE_FOLLOWING_1_NUMBERS_ARE_ALREADY_IN_INVENTORY_2;
APP_DEPRECATION_WARN	The record <record name> is deprecated. A new required field, <field name>, has been added to this record. We are setting this field to the default value of <default value>. The deprecation period will end on <end date>. The current version of your client will be obsolete at the end of the deprecation period, please upgrade to the latest version.
APP_DEPRECATION_WARN	The record <record name> is deprecated. The <field name> field is now required. We are setting this field to the default value of <default value>. The deprecation period will end on <end date>. The current version of your client will be obsolete at the end of the deprecation period, please upgrade to the latest version.
APP_DEPRECATION_WARN	The record <record name> is deprecated. The type of the <field name> field has changed to <new field type>. We are mapping the value of this field to <mapped value>. The deprecation period will end on <end date>. The current version of your client will be obsolete at the end of the deprecation period, please upgrade to the latest version.
APP_DEPRECATION_WARN	The record <record name> is deprecated. The <field name> field has been renamed to <new field name>. We are mapping the old name to the new name. The deprecation period will end on <end date>. The current version of your client will be obsolete at the end of the deprecation period, please upgrade to the latest version.
APP_DEPRECATION_WARN	The record <record name> is deprecated. The <field name> field has been removed. The deprecation period will end on <end date>. The current version of your client will be obsolete at the end of the deprecation period, please upgrade to the latest version.
APP_DEPRECATION_WARN	The <record name> record is deprecated. It has been renamed to <new record name>. The deprecation period will end on <end date>. The current version of your client will be obsolete at the end of the deprecation period, please upgrade to the latest version.
APP_DEPRECATION_WARN	The <record name> record is deprecated. It has been removed. The deprecation period will end on <end date>. The current version of your client will be obsolete at the end of the deprecation period, please upgrade to the latest version.
AREA_CODE_REQD	PLEASE_INCLUDE_AN_AREA_CODE_WITH_THE_PHONE_NUMBER;

Error Code Returned	Long Description or Message
ATTACH_SIZE_EXCEEDED	YOU_HAVE_EXCEEDED_THE_MAXIMUM_ATTACHMENTS_SIZE_OF_10_MB_PLEASE_REMOVE_ONE_OR_MORE_ATTACHMENTS_AND_TRY_AGAIN;
AUTO_NUM_UPDATE_DISALLOWED	WE_CURRENTLY_DO_NOT_SUPPORT_AN_AUTOMATIC_NUMBERING_UPDATE_OF_MORE_THAN_1_2_RECORDS;
BASE_CRNCY_REQD	YOU_MAY_NOT_DELETE_YOU_BASE_CURRENCY;
BILL_PAY_STATUS_UNAVAILABLE	VIEW_ONLINE_BILL_PAY_STATUS_INFORMATION_IS_CURRENTLY_NOT_AVAILABLE_PLEASE_TRY_AGAIN_IN_A_FEW_MINUTES;
BILL_PAY_STATUS_UNAVAILABLE	VIEW_ONLINE_BILL_PAY_STATUS_IS_NOT_AVAILABLE_UNTIL_YOUR_BILL_PAY_REGISTRATION_IS_COMPLETE;
BILL_PMTS_MADE_FROM_ACCT_ONLY	YOUR_PAYMENT_HAS_BEEN_RECORDED_BUT_ONLINE_BILL_PAYMENTS_CAN_ONLY_BE_MADE_FROM_THE_ACCOUNT;
BILLING_ISSUES	YOUR_ACCOUNT_HAS_BEEN_LOCKED_DUE_TO_BILLING_ISSUES_YOU_MUST_CALL_YOUR_SALES_REPRESENTATIVE_AT_1800SMALLBIZ_FOR_FURTHER_ASSISTANCE;
BILLING_ISSUES	YOUR_ACCOUNT_HAS_NOT_BEEN_FULLY_PAID_FOR_PLEASE_LOG_IN_TO_YOUR_ACCOUNT_AND_FOLLOW_THE_BILLING_PROCESS_OR_CONTACT_YOUR_ACCOUNT_MANAGER;
CALENDAR_PREFS_REQD	SET_UP_1_CALENDAR_PREFERENCES_FIRST;
CALENDAR_PREFS_REQD	SET_UP_CALENDAR_PREFERENCES_FIRST;
CAMPAGIN_ALREADY_EXECUTED	YOU_CANNOT_DELETE_EMAIL_CAMPAIGNS_THAT_HAVE_ALREADY_BEEN_EXECUTED;
CAMPAIGN_IN_USE	YOU_CANNOT_DELETE_A_CAMPAIGN_EVENT_THAT_ALREADY_HAS_ACTIVITY;
CAMPAIGN_SET_UP_REQD	THE_FOLLOWING_STEPS_NEED_TO_BE_PERFORMED_BEFORE_A_CAMPAIGN_CAN_BE_CREATED;
CANT_CREATE_NON_UNIQUE_RECORD	WS_ERROR_CREATING_NON_UNIQUE_RECORD;
CANT_DELETE_CHILD_RECORD_FOUND	WS_CHILD_RECORD_FOUND_IN_DELETE;
CANT_DELETE_CHILD_RECORDS_EXIST	THIS_RECORD_CAN_NOT_BE_DELETED_BECAUSE_IT_HAS_CHILD_RECORDS;
CANT_DELETE_CUST_PLACEHOLDER	YOU_CANNOT_DELETE_THE_ANONYMOUS_CUSTOMER_PLACEHOLDER_IF_YOU_MUST_DELETE_THIS_RECORD_FIRST_GO_TO_THE_A_HREFAPPSITESETUPADMINNLSET_UP_WEB_SITEA_PAGE_AND_DESELECT_THE_ANONYMOUS_CUSTOMER_PLACEHOLDER;
CANT_DELETE_LEGACY_CATEGORY	LEGACY_CATEGORY_CANNOT_BE_REMOVED;
CANT_DELETE_STATUS_TYPE	YOU_CANNOT_DELETE_THE_ONLY_STATUS_OF_TYPE_1;
CANT_ESTABLISH_LINK	UNABLE_TO_ESTABLISH_LINK_WITH_1;
CANT_SEND_EMAIL	UNABLE_TO_SEND_NOTIFICATION_EMAIL;
CANT_SEND_EMAIL	UNABLE_TO_SEND_NOTIFICATION_EMAIL_TO_SUPPORT_REP;

Error Code Returned	Long Description or Message
CANT_SET_CLOSE_DATE	UNABLE_TO_SET_EXPECTED_CLOSE_DATE_OF_PROSPECTLEAD_BASED_ON_CURRENT_ESTIMATESOPPORTUNITIES;
CANT_SET_STATUS	UNABLE_TO_SET_STATUS_OF_PROSPECTLEAD_BASED_ON_CURRENT_ESTIMATES;
CANT_SWITCH_ROLES_FROM_LOGIN	ROLE_SWITCHING_IS_NOT_ALLOWED_FROM_THIS_LOGIN;
CANT_UPDATE_ATTACHMENT	Updating an attachment in this context is not supported. You can explicitly update the file through an update operation on the File record.
CANT_UPDATE_STATUS_TYPE	YOU_CANNOT_UPDATE_THE_ONLY_STATUS_OF_TYPE_1;
CASE_ALREADY_ASSIGNED	THIS_CASE_CANNOT_BE_GRABBED_BECAUSE_IT_IS_ALREADY_ASSIGNED_TO_ANOTHER_REP__TO_VIEW_THE_CASE_GO_BACK_AND_CLICK_ON_THE_CASE_NUMBER;
CASE_NOT_GROUP_MEMBER	_1_THIS_CASE_RECORD_DOES_NOT_BELONG_TO_YOUR_GROUP;
CASH_SALE_EDIT_DISALLWD	THIS_CASH_SALE_CANNOT_BE_EDITED_WHILE_IT_HAS_AN_AUTOMATED_CLEARING_HOUSE_TRANSMISSION_IN_PROCESSDTRTRTD_CLASSTEXTNBSPDTRTRTD_CLASSTEXTNBSPD_VIEW_THE_STATUS_OF_CASH_SALES_WITH_ACH_TRANSMISSIONS_GO_TO_TRANSACTIONS__VIEW_ELECTRONIC_FUNDS_TRANSFER_STATUS;
CC_ALREADY_SAVED	THAT_CREDIT_CARD_IS_ALREADY_SAVED__PLEASE_USE_THE_SAVED_CREDIT_CARD;
CC_EMAIL_ADDRESS_REQD	PLEASE_GO_BACK_AND_PROVIDE_AN_EMAIL_ADDRESS_TO_CC_STORE_ORDERS_TO;
CC_NUM_REQD	PLEASE_PROVIDE_A_CREDIT_CARD_NUMBER;
COMMSSN_ALREADY_CALCLTD	A_PLANS_SCHEDULES_OR_PAST_SALES_REP_ASSIGNMENTS_CANNOT_BE_MODIFIED_ONCE_COMMISSIONS_AGAINST_THE_PLAN_HAVE_BEEN_CALCULATED;
COMMSSN_FEATURE_DISABLED	YOU_HAVE_NOT_ENABLED_THE_COMMISSIONS_FEATURE;
COMMSSN_PAYROLL_ITEM_REQD	A_COMMISSION_PAYROLL_ITEM_MUST_BE_ADDED_FOR_EACH_EMPLOYEE_TO_BE_PROCESSED_THROUGH_PAYROLL;
COMPANION_PROP_REQD	ERROR_ITEMS_DO_NOT_HAVE_COMPANION_PROPERTY_COLUMN_0;
COMPANY_RCRD_DELETED	THE_COMPANY_YOU_TRY_TO_ATTACH_THE_CONTACT_TO_HAS_BEEN_DELETED_OR_MERGED;
CONSLD_PRNT_AND_CHILD_DISALLWD	A_COMPANY_CAN_BE_A_CONSOLIDATED_CHILD_OR_A_CONSOLIDATED_PARENT_BUT_NOT_BOTH;
CONTACT_NOT_GROUP_MEMBER	_1_THIS_CONTACT_DOES_NOT_BELONG_TO_YOUR_GROUP;
COOKIES_DISABLED	YOU_HAVE_DISABLED_COOKIES_FROM_BEING_STORED_ON_YOUR_COMPUTER_OR_TURNED_OFF_PERSESSION_COOKIES_PLEASE_ENABLE_THIS_FEATURE_AND_TRY_AGAIN;
CRNCY_NOT_UPDATED	THE_FOLLOWING_CURRENCIES_WERE_NOT_UPDATED;
CRNCY_RCRD_DELETED	THIS_CURRENCY_RECORD_HAS_BEEN_DELETED__YOU_CAN_CREATE_A_NEW_CURRENCY_RECORD_AT_LISTS__CURRENCIES;
CSTM_FIELD_KEY_REQD	WS_CUSTOM_FIELD_KEY_IS_MISSING;

Error Code Returned	Long Description or Message
CSTM_FIELD_VALUE_REQD	WS_CUSTOM_FIELD_VALUE_IS_MISSING;
CUST_ACCESS_FEATURE_DISABLED	CANT_OPEN_STORE_FOR_1__THIS_COMPANY_DOES_NOT_HAVE_THE_B CUSTOMER_ACCESSB_FEATURE_ENABLED__THIS_FEATURE_IS_REQUIRE D_FOR_CUSTOMERS_TO_BE_ABLE_TO_REGISTER_CHECK_OUT_AND_LO G_IN_TO_THE_SCORE;
CUST_LEAD_NOT_GROUP_MEMBER	_1_THIS_CUSTOMER_OR_LEAD_DOES_NOT_BELONG_TO_YOUR_GROUP;
DATA_MUST_BE_UNIQUE	THE_UPDATE_FAILED_BECAUSE_EVERY_ENTRY_IN_THIS_COLUMN_MUS T_BE_UNIQUE;
DATA_REQD	WS_PROVIDE_A_PROPER_VALUE_FOR_THE_REQUIRED_FIELD;
DATE_EXPECTED	YOU_ENTERED_1_INTO_A_FIELD_WHERE_A_CALENDAR_DATE_WAS_EX PECTEDPLEASE_GO_BACK_AND_CHANGE_THIS_VALUE_TO_THE_CORR ECT_DATE;
DATE_PARAM_REQD	MISSING_DATE_PARAMETER;
DEFAULT_TYPE_DELETE_DISALLWD	YOU_CANNOT_DELETE_DEFAULT_TYPES;
DEPT_IN_USE	YOUR_CLASSES_CANNOT_BE_CONVERTED_TO_DEPARTMENTS_BECAUS E_YOUR_EXISTING_DEPARTMENT_RECORDS_ARE_REFERRED_TO_BY_TR ANSACTIONS_OR_OTHER_RECORDS_THESE_DEPARTMENT_RECORDS_CA NNOT_BE_OVERWRITTEN;
DISALLWD_IP_ADDRESS	THE_SPECIFIED_IP_ADDRESS_RULES_MUST_ALLOW_THE_LOGIN_OF_YO UR_CURRENT_IP_ADDRESS__YOUR_CURRENT_IP_ADDRESS_IS_1__FOR _INFORMATION_ON_ENTERING_IP_ADDRESS_RULES_CLICK_HELP_AT_T HE_TOP_OF_THE_PAGE;
DISTRIB_REQD_ONE_DAY_BEFORE	ALL_ITEMS_MUST_BE_DISTRIBUTED_AT_LEAST_ONE_DAY_BEFORE_THE Y_MAY_BE_TRANSFERRED;
DFRNT_SWAP_PRICE_LEVELS_REQD	PLEASE_SELECT_DIFFERENT_PRICE_LEVELS_TO_SWAP_PRICES;
DUE_DATE_BFORE_START_DATE	DUE_DATE_OCCURS_BEFORE_START_DATE;
DUP_PAYROLL_ITEM	THERE_IS_ALREADY_A_PAYROLL_ITEM_NAMED_1;
DUP_TRACKING_NUM	YOU_ENTERED_THE_FOLLOWING_TRACKING_NUMBER_TWICE;
DUPLICATE_INVENTORY_NUM	DUPLICATE_INVENTORY_NUMBER_FOUND_IN_ENTRY_1;
DUPLICATE_INVENTORY_NUM	DUPLICATE_INVENTORY_NUMBER_FOUND_ON_DIFFERENT_LINES_OF_T RANSACTION;
DUPLICATE_KEYS	WS_DUPLICATED_KEYS;
DUPLICATE_NAME_FOR_PRD	PLEASE_CHOOSE_A_DIFFERENT_PERIOD_NAME_1_IS_ALREADY_TAKEN;
DUPLICATE_NAME_FOR_ROLE	PLEASE_CHOOSE_A_DIFFERENT_ROLE_NAME_1_IS_ALREADY_TAKEN;
DUPLICATE_USER_NAME	A_USER_WITH_THIS_NAME_ALREADY_EXISTS;
EMAIL_ADDRS_REQD	PLEASE_ENTER_YOUR_EMAIL_ADDRESS;
EMAIL_ADDRS_REQD_TO_NOTIFY	PLEASE_ENTER_AN_EMAIL_ADDRESS_FOR_THIS_COMPANY_A_NOTIFICA TION_EMAIL_WILL_BE_SENT_WHEN_THIS_CASE_RECORD_IS_SAVED;

Error Code Returned	Long Description or Message
EMAIL_ADDRS_REQD_TO_NOTIFY	THE_RECIPIENT_YOU_ARE_SENDING_THIS_EMAIL_TO_DOES_NOT_HAVE_AN_EMAIL_ADDRESS_PLEASE_ENTER_ONE_AND_TRY_AGAIN;
EMAIL_REQD	YOU_MUST_ENTER_A_VALID_EMAIL_ADDRESS_IN_ORDER_TO_EMAIL_THE_TRANSACTION;
EMPL_IN_USE	YOU_CANT_DELETE_THIS_EMPLOYEE_AS_COMMISSIONS_HAVE_BEEN_CALCULATED_FOR_THIS_EMPLOYEE;
EMPL_IN_USE	YOU_CANT_DELETE_THIS_EMPLOYEE_AS_IT_IS_OR_HAS_BEEN_REFERENCED_BY_OTHER_EMPLOYEES_AS_A_SUPERVISOR;
ERROR_PRCSSNG_TRANS	THERE_WERE_ERRORS_PROCESSING_THE_SELECTED_TRANSACTIONS_PLEASE_PROCESS_THEM_INDIVIDUALLY_FOR_MORE_INFORMATION;
EVENT_ID_NOT_FOUND	EVENT_ID_NOT_FOUND;
EXPIRED_SEARCH_CRITERIA	YOUR_SEARCH_CRITERIA_EXPIRED_THE_CRITERIA_FOR_A_GIVEN_SEARCH_GENERALLY_EXPIRE_AFTER_15_MINUTES_OF_INACTIVITY_CLICK_A_HISTORY_ONCLICKHISTORYGO_HEREA_TO_GO_BACK_TO_THE_SEARCH_DEFINITION_PAGE_AND_RESUBMIT_YOUR_SEARCH;
FAX_NUM_REQD	YOU_MUST_ENTER_A_FAX_NUMBER;
FAX_NUM_REQD	YOU_MUST_ENTER_A_FAX_NUMBER_FOR_THIS_RECIPIENT_BEFORE_PERFORMING_A_FAX_MERGE_OPERATION;
FAX_NUM_REQD	YOU_MUST_ENTER_A_VALID_FAX_NUMBER_IN_ORDER_TO_FAX_THE_TRANSACTION;
FEATURE_DISABLED	THE_FEATURE_1_REQUIRED_TO_ACCESS_THIS_PAGE_IS_NOT_ENABLED_IN_THIS_ACCOUNT;
FEATURE_UNAVAILABLE	THE_1_FEATURE_IS_NOT_AVAILABLE_TO_YOUR_COMPANY;
FED_WITHHOLDING_REQD	YOUR_EMPLOYEE_RECORD_DOES_NOT_HAVE_CURRENT_FEDERAL_WITHHOLDING_INFORMATION_PLEASE_CONTACT_YOUR_SUPERVISOR_TO_SET_UP_YOUR_RECORD_WITH_THE_APPROPRIATE_INFORMATION_P;
FIELD_CALL_DATE_REQD	MISSING_REQUIRED_FIELD_CALL_DATE;
FIELD_DEFN_REQD	FIELD_DEFINITION_NOT_FOUND;
FIELD_REQD	YOU_MUST_FIRST_SELECT_A_FIELD;
FILE_NOT_FOUND	FILEMEDIA_ITEM_1_NOT_FOUND;
FILE_REQD	YOU_MUST_UPLOAD_A_FILE_BEFORE_CREATING_THIS_MEDIA_ITEM;
FILTER_BY_AMT_REQD	PLEASE_ENTER_AN_AMOUNT_TO_FILTER_BY;
FIRST_LAST_NAMES_REQD	PLEASE_ENTER_BOTH_YOUR_FIRST_AND_LAST_NAME;
FORM_RESUBMISSION_REQD	YOU_HAVE_LOGGED_IN_TO_A_DIFFERENT_USER_SINCE_YOU_NAVIGATED_TO_THIS_FORM__YOU_MUST_RESUBMIT_THIS_FORM_AS_THE_NEW_USER;
FULL_DISTRIB_REQD	YOU_MUST_FULLY_DISTRIBUTE_ALL_1_NUMBERS_FOR_1_NUMBERED_ITEMS;
FULL_USERS_REQD_TO_INTEGRATE	ONLY_FULL_1_USERS_CAN_INTEGRATE_WITH_PARTNERS;
GETALL_RCRD_TYPE_REQD	WS_MISSING_GETALL_RECORD_TYPE;

Error Code Returned	Long Description or Message
GROUP_DSNT_EXIST	THAT_GROUP_DOES_NOT_EXIST;
GROUP_REQD	YOU_CANNOT_PERFORM_A_BULK_MERGE_OPERATION_WITH_AN_EMPTY_GROUP;
ILLEGAL_PERIOD_STRUCTURE	ILLEGAL_PERIOD_STRUCTURE_DATE_1_IS_IN_MULTIPLE_PERIODS;
INACTIVE_RCRD_FOR_ROLE	THE_RECORD_FOR_THIS_ROLE_HAS_BEEN_MADE_INACTIVE;
INCRCT_ORD_INFO	THE_ORDER_CONTAINS_INCORRECT_INFORMATION_AND_WAS_NOT_PLACED;
INSUFcnt_NUM_PRDS_FOR_REV_REC	NOT_ENOUGH_ACCOUNTING_PERIODS_IN_RANGE_SPECIFIED_FOR_REVENUE_RECOGNITION;
INSUFcnt_OPEN_PRDS_FOR_REV_REC	NOT_ENOUGH_OPEN_ACCOUNTING_PERIODS_AVAILABLE_FOR_REVENUE_RECOGNITION;
INCRCT_ORD_INFO	THE_ORDER_CONTAINS_INCORRECT_INFORMATION_AND_WAS_NOT_PLACED;
INSUFFICIENT_CHARS_IN_SEARCH	GLOBAL_SEARCHES_MUST_CONTAIN_AT_LEAST_THREE_CHARACTERS_TO_PREVENT_EXCESSIVE_MATCHES;
INSUFFICIENT_PERMISSION	YOU_MUST_HAVE_TRANSACTIONS_FULFILL_SALES_ORDERS_PERMISSION_TO_FULFILL_SALES_ORDERS;
INSUFFICIENT_PERMISSION	YOU_MUST_HAVE_EITHER_TRANSACTIONS_INVOICE_OR_TRANSACTIONS_CASH_SALE_PERMISSION_TO_BILL_SALES_ORDERS;
INSUFFICIENT_PERMISSION	YOU_NEED_EMPLOYEE_ACCESS_IN_ORDER_TO_DELETE_THIS_RECORD;
INSUFFICIENT_PERMISSION	FOR_SECURITY_REASONS_ONLY_AN_ADMINISTRATOR_IS_ALLOWED_TO_EDIT_AN_ADMINISTRATOR_RECORD;
INSUFFICIENT_PERMISSION	THIS_ORDER_HAS_BEEN_PARTIALLY_OR_FULLY_PROCESSED_AND_MAY_NOT_BE_EDITED_BY_A_USER_WITHOUT_PERMISSION_TO_APPROVE_SALES_ORDERS;
INSUFFICIENT_PERMISSION	YOU_DO_NOT_HAVE_ACCESS_TO_THE_ACTIVITY_HISTORY_FOR_THAT_RECORD;
INSUFFICIENT_PERMISSION	YOU_DO_NOT_HAVE_ACCESS_TO_THE_MEDIA_ITEM YOU_SELECTED;
INSUFFICIENT_PERMISSION	YOU_DO_NOT_HAVE_PERMISSION_TO_EMAIL_TRANSACTIONS;
INSUFFICIENT_PERMISSION	YOU_MUST_HAVE_EITHER_TRANSACTIONS_INVOICE_OR_TRANSACTIONS_CASH_SALE_PERMISSION_TO_FULFILL_SALES_ORDERS;
INSUFFICIENT_PERMISSION	YOU_ARE_NOT_ALLOWED_TO_APPROVE_YOUR_OWN_TRANSACTIONS;
INSUFFICIENT_PERMISSION	_1_THE_RESTRICTIONS_ON_YOUR_ROLE_DENY_YOU_ACCESS_TO_THIS_RECORD;
INSUFFICIENT_PERMISSION	_1_THE_2_RESTRICTIONS_ON_YOUR_ROLE_DENY_YOU_ACCESS_TO_THIS_RECORD;
INSUFFICIENT_PERMISSION	_1_THE_2_RESTRICTIONS_ON_YOUR_ROLE_PREVENT_YOU_FROM_SEEING_THIS_RECORD;
INSUFFICIENT_PERMISSION	INSUFFICIENT_PRIVILEGES;
INSUFFICIENT_PERMISSION	YOUR_ROLE_DOES_NOT_HAVE_PERMISSION_TO_PROVISION_ACCOUNTS;

Error Code Returned	Long Description or Message
INSUFFICIENT_PERMISSION	USER_PERMISSION_LEVEL_COULD_NOT_BE_ESTABLISHED;
INSUFFICIENT_PERMISSION	YOU_DO_NOT_HAVE_PERMISSION_TO_PERFORM_THIS_OPERATION;
INSUFFICIENT_PERMISSION	YOU_DO_NOT_HAVE_PERMISSION_TO_VIEW_THIS_PAGE;
INSUFFICIENT_PERMISSION	YOU_DO_NOT_HAVE_PRIVILEGES_TO_PERFORM_THIS_ACTION;
INSUFFICIENT_PERMISSION	YOU_DO_NOT_HAVE_PRIVILEGES_TO_PERFORM_THIS_OPERATION;
INSUFFICIENT_PERMISSION	YOU_DO_NOT_HAVE_PRIVILEGES_TO_USE_THIS_PAGE;
INSUFFICIENT_PERMISSION	YOU_DO_NOT_HAVE_PRIVILEGES_TO_VIEW_THIS_PAGE;
INSUFFICIENT_PERMISSION	_1_THE_CUSTOMER_RESTRICTIONS_ON_YOUR_PARTNER_ROLE_PREVENT_YOU_FROM_SEEING_THIS_RECORD;
INSUFFICIENT_PERMISSION	_1_YOU_NEED_1_THE_2_PERMISSION_TO_ACCESS_THIS_PAGE_PLEASE_CONTACT_YOUR_ACCOUNT_ADMINISTRATOR;
INSUFFICIENT_PERMISSION	PERMISSION_ERROR_YOU_MAY_NOT_EDIT_THIS_ROLE;
INSUFFICIENT_PERMISSION	PERMISSION_VIOLATION_PARTNERS_DO_NOT_HAVE_ACCESS_TO_THIS_REPORT;
INSUFFICIENT_PERMISSION	PERMISSION_VIOLATION_PARTNERS_MAY_NOT_DELETE_SAVED_REPORTS;
INSUFFICIENT_PERMISSION	PERMISSION_VIOLATION_YOU_CANNOT_DELETE_SAVED_REPORTS_NOT_CREATED_BY_YOURSELF;
INSUFFICIENT_PERMISSION	WS_NO_PERMISSIONS_TO_SET_VALUE
INTEGER_REQD_FOR_QTY	QUANTITY_MUST_BE_AN_INTEGER_FOR_NUMBERED_ITEMS;
INVALID_ACCT	INVALID_LOGIN_NO_SUCH_ACCOUNT;
INVALID_ACCT	WS_INVALID_ACCOUNT_NUMBER;
INVALID_ADJUSTMENT_ACCT	THE_ACCOUNT_YOU_SELECTED_IN_ADJUSTMENT_ACCOUNT_IS_THE_SAME_AS_THE_ASSET_ACCOUNT_FOR_ONE_OF_THE_ITEMS_YOU_ARE_ADJUSTING_PLEASE_GO_BACK_AND_CHANGE_THE_ACCOUNT_NORMALLY_THE_ADJUSTMENT_ACCOUNT_WOULD_BE_AN_EXPENSE_ACCOUNT;
INVALID_AUTHORIZATION	CORRECT_AUTHORIZATION_EXCEPTION;
INVALID_BALANCE_RANGE	YOUR_BALANCE_IS_NOT_WITHIN_THE_ALLOWED_RANGE;
INVALID_BUG_NUM	BUG_NUMBER_SPECIFIED_WAS_INCORRECT__1_ISNT_A_NUMBER;
INVALID_CAMPAIGN_GROUP_SIZE	WHILE_IN_1_YOU_CAN_ONLY_SEND_2_EMAILS_PER_CAMPAIGN_EVENT__PLEASE_MODIFY_ONE_OR_MORE_OF_YOUR_TARGET_GROUPS_TO_CONTAIN_2_MEMBERS_OR_LESS_ALL_CAMPAIGN_EMAILS_WILL_BE_SENT_TO_YOUR_1_LOGIN_EMAIL_ADDRESS;
INVALID_CATGRY_TAX_AGENCY_REQ	A_VENDOR_MUST_BE_CREATED_IN_A_CATEOGRY_WITH_THE_TAX_AGENCY_CHECKBOX_CHECKED;
INVALID_CC_EMAIL_ADDRESS	THE_EMAIL_ADDRESS_TO_CC_STORE_ORDERS_TO_IS_INVALID__PLEASE_GO_BACK_AND_CORRECT_IT;
INVALID_CC_NUM	CREDIT_CARD_NUMBER_IS_NOT_VALID__PLEASE_CHECK_THAT_ALL_DIGITS_WERE_ENTERED_CORRECTLY;
INVALID_CC_NUM	CREDIT_CARD_NUMBER_MUST_CONTAIN_ONLY_DIGITS;

Error Code Returned	Long Description or Message
INVALID_CC_NUM	CREDIT_CARD_NUMBERS_MUST_CONTAIN_BETWEEN_13_AND_16_DIGITS;
INVALID_CHARS_IN_EMAIL	EMAIL_ADDRESS_CONTAINS_INVALID_CHARACTERS;
INVALID_CHARS_IN_NAME	THE_FROM_NAME_FIELD_CANNOT_CONTAIN_APOSTROPHES_QUOTATION_MARKS_COMMAS_OR_GREATER_THAN_OR_LESS_THAN_SIGNS;
INVALID_CHARS_IN_NAME	YOU_CANNOT_USE_THE_COLON_CHARACTER_IN_THE_TOPIC_NAME__PLEASE_REMOVE_IT_;
INVALID_CHARS_IN_PARAM_FIELD	THE_ADDITIONAL_PARAMETERS_FIELD_CAN_NOT_CONTAIN_ANY_OF_THE_FOLLOWING_CHARACTERS__PLEASE_REMOVE_THEM_AND_TRY_AGAIN;
INVALID_CHARS_IN_URL	SPACES_ARE_NOT_ALLOWED_IN_THE_1URLPEXAMPLES_OF_A_VALID_1_URL_AREBRBHTTPWWWMYDOMAINCOMIMAGEGIFBNBSPNBSPOBNBSPNBSPBHTTTPSONETWOORGUSERNAMETESTJPG;
INVALID_CHARS_IN_URL	THE_URL_COMPONENT_YOU_HAVE_CHOSEN_CONTAINS_A_SPACE_OR_ONE_OF_THE_FOLLOWING_PROHIBITED_CHARACTER__PLEASE_REMOVE_THEM_AND_TRY_AGAIN;
INVALID_COSTING_METHOD	SERIAL_AND_LOT_ARE_THE_ONLY_COSTING_METHODS_THAT_MAY_BE_PASSED_AS_PARAMETERS_TO_THIS_PAGE;
INVALID_CREDENTIALS	NCEL_AUTHORIZATION_EXCEPTION;
INVALID_CSTM_FIELD_REF	WS_INVALID_CUSTOM_FIELD_REF;
INVALID_CSTM_RCRD_TYPE_KEY	WS_INVALID_CUSTOM_RECORD_TYPE_KEY;
INVALID_DATA	Error in record number <nskey>: Invalid field value <field>. Please refer to the XSD for enumerated list of valid field values."
INVALID_DATE_RANGE	INVALID_DATE_RANGENTHE_EVENT_1_START_TIME_2_MUST_BE_EARLIER_THAN_THE_END_TIME_3;
INVALID_DATE_RANGE	THE_DATE_RANGE_YOU_SPECIFIED_DOES_NOT_FALL_INSIDE_THAT_OF_THE_PARENT_PERIOD;
INVALID_EMAIL	EMAIL_ADDRESS_IS_NOT_VALID;
INVALID_EMAIL	YOU_HAVE_ENTERED_AN_INVALID_EMAIL_ADDRESS__PLEASE_TRY_AGAIN;
INVALID_EMAIL	YOUR_EMAIL_OR_CODE_IS_INVALID__PLEASE_TRY_AGAIN;
INVALID_END_DATE	EVENT_1_RECURRENCE_END_DATE_IS_INVALID;
INVALID_END_TIME	INVALID_END_TIME;
INVALID_FILE	VERIFY_THAT_YOU_HAVE_A_VALID_FILE_TO_UPLOAD;
INVALID_FILE_TYP	INVALID_FILE_TYPE_FILE_IS_NOT_A_COMPRESSED_ZIP_FILE;
INVALID_FILE_TYP	INVALID_FILE_TYPE_FILE_IS_NOT_A_COMPRESSEDZIP_FILE;
INVALID_FORMAT_IN_PARAM_FIELD	THE_ADDITIONAL_PARAMETERS_FIELD_IS_NOT_FORMATTED_CORRECTLY__PLEASE_REFORMAT_AND_TRY_AGAIN;
INVALID_FROM_DATE	INVALID_FROM_DATE;
INVALID_FROM_TIME	INVALID_FROM_TIME;

Error Code Returned	Long Description or Message
INVALID_GROUP_SIZE	YOU_CANNOT_PERFORM_A_BULK_MERGE_OPERATION_WITH_A_GROUP_LARGER_THAN_500_RECORDS;
INVALID_GST_PST_AGENCIES	THE_GST_OR_PST_AGENCIES_ARE_NOT_VALID_PLEASE_REVIEW_YOUR_COMPANY_PREFERENCES;
INVALID_INVENTORY_NUM	INVALID_SET_OF_INVENTORY_NUMBERS_VALUES_MUST_BE_SEPARATED_BY_COMMAS_SPACES_TABS_OR_LINE_FEEDS;
INVALID_IP_ADDRESS_RULE	THE_FOLLOWING_IP_ADDRESS_RULE_IS_NOT_VALID_1;
INVALID_KEY_OR_REF	WS_INVALID_KEY; Note: Users encounter this error when they provide an invalid key for a field of type "RecordRef" in <i>add</i> or <i>update</i> operations.
INVALID_KEY_OR_REF	WS_INVALID_REFERENCE_KEY_1;
INVALID_KEY_OR_REF	WS_INVALID_REFERENCE_KEY_2;
INVALID_LOGIN	INVALID_LOGIN_ONLINE_FORM_ACCESS_IS_DISABLED;
INVALID_LOGIN	INVALID_LOGIN_SUPPLIER_ACCESS_IS_DISABLED;
INVALID_LOGIN_ATTEMPT	INVALID_LOGIN_ATTEMPT;
INVALID_LOGIN_CREDENTIALS	A_PROBLEM_OCCURED_VERIFYING_THE_PRESENTED_EMAIL_ADDRESS_PASSWORD_ROLENAME_OR_ACCOUNT_NUMBER_PLEASE_VERIFY_THESE_PIECES_OF_INFORMATION_AND_TRY_AGAIN;
INVALID_LOGIN_CREDENTIALS	YOU_HAVE_ENTERED_AN_INVALID_EMAIL_ADDRESS_OR_ACCOUNT_NUMBER_PLEASE_TRY_AGAIN;
INVALID_LOGIN_CREDENTIALS	YOU_HAVE_ENTERED_AN_INVALID_EMAIL_ADDRESS_OR_PASSWORD_PLEASE_TRY_AGAIN;
INVALID_LOGIN_CREDENTIALS	YOU_HAVE_ENTERED_AN_INVALID_LOGIN_PASSWORD__PLEASE_TRY_AGAIN;
INVALID_LOGIN_CREDENTIALS	YOU_HAVE_ENTERED_AN_INVALID_PASSWORD__PLEASE_TRY_AGAIN;
INVALID_LOGIN_CREDENTIALS	Invalid Login. The email address, account number and role you have specified match more than one employee. Please contact your administrator to correct the problem.
INVALID_LOGIN_IP	INVALID_LOGIN_IP_ADDRESS_DOES_NOT_MATCH_ANY_OF_THE_IPADDRESS_RULES_SPECIFIED_FOR_THIS_ENTITY;
INVALID_LOT_NUM_FORMAT	LOT_NUMBERS_MUST_BE_ENTERED_USING_THIS_FORMAT;
INVALID_MARKUP_DISCOUNT	MARKUPDISCOUNT__MUST_BE_BETWEEN_999_AND_999;
INVALID_MEMRZD_TRANS	A_MEMORIZED_TRANSACTION_MAY_NOT_CONTAIN_ANY_SERIAL_OR_LOT_NUMBERS;
INVALID_NUMBER	INVALID_INTEGER_1;
INVALID_NUMBER	INVALID_NUMBER_1;
INVALID_ORD_STATUS	THIS_ORDER_HAS_BEEN_PARTIALLY_OR_FULLY_PROCESSED_AND_MAY_NOT_BE_RESET_TO_PENDING_APPROVAL;
INVALID_POST	INVALID_POST;
INVALID_PST_TAX_VALUE	PST_TAX_VALUE_IS_NOT_A_VALID_NUMBER_1;

Error Code Returned	Long Description or Message
INVALID_PSWD	YOUVE_USED_THAT_PASSWORD_BEFORE__PLEASE_CHOOSE_A_NEW_PASSWORD;
INVALID_PSWD	PASSWORD_MUST_BE_AT_LEAST_6_CHARACTERS_LONG;
INVALID_PSWD	PASSWORD_MUST_BE_AT_LEAST_6_CHARACTERS_LONG_AND_CONTAIN_AT_LEAST_ONE_NUMBER_OR_SPECIAL_CHARACTER;
INVALID_PSWD	PASSWORD_MUST_CONTAIN_AT_LEAST_ONE_LETTER_AZ;
INVALID_PSWD	PASSWORD_MUST_CONTAIN_AT_LEAST_ONE_NUMBER_OR_SPECIAL_CHARACTER;
INVALID_PSWD	PASSWORD_PROVIDED_DOES_NOT_MATCH_SYSTEM_PASSWORD;
INVALID_PSWD	YOUR_PASSWORD_MUST_BE_AT_LEAST_6_CHARACTERS;
INVALID_PSWD	THE_CURRENT_PASSWORD_YOU_SUPPLIED_IS_INCORRECT;
INVALID_PSWD	YOUR_NEW_PASSWORD_MUST_BE_AT_LEAST_6_CHARACTERS_CONTAIN_AT_LEAST_ONE_NONLETTER_AND_BE_SUBSTANTIALLY_DIFFERENT_FROM_THE_CURRENT_PASSWORD;
INVALID_PSWD_ILLEGAL_CHAR	PASSWORD_CONTAINS_AN_ILLEGAL_CHARACTER;
INVALID_QUANTITY	SERIAL_AND_LOT_NUMBER_QUANTITIES_MUST_BE_POSITIVE;
INVALID_RCRD_TYPE	WS_INVALID_RECORD_TYPE;
INVALID_REFFERER_EMAIL	THE_REFFERER_EMAIL_ADDRESS_YOU_HAVE_ENTERED_IS_NOT_VALID__PLEASE_TRY_AGAIN;
INVALID_ROLE	WS_ROLE_IS_INVALID;
INVALID_ROLE	YOU_HAVE_ENTERED_AN_INVALID_ROLE_ID_PLEASE_TRY_AGAIN;
INVALID_ROLE_FOR_EVENT	YOU_SEEM_TO_HAVE_BEEN_INVITED_TO_THIS_EVENT_IN_A_DIFFERENT_ROLE_PLEASE_CHANGE_YOUR_ROLE_TO_VIEW_THE_EVENT;
INVALID_RQST_CONTACTS_EXIST	IT_HAS_ASSOCIATED_PRIMARY_CONTACTS;
INVALID_RQST_PARENT_REQD	IT_HAS_ASSOCIATED_CONTACT_RECORDS_THAT_WOULD_BE_LEFT_WITH_NO_PARENT_COMPANY;
INVALID_RQST_SBCUST_JOBS_EXIST	IT_HAS_ASSOCIATED_SUBCUSTOMERS_OR_JOBS;
INVALID_SCHDUL_FORMAT	TO_CREATE_A_VALID_SCHEDULE_PLEASE_ENTER_THE_BRACKET_VALUES_IN_ASCENDING_ORDERS_WITHOUT_GAPS;
INVALID_SEARCH	THAT_SEARCH_OR_MASS_UPDATE_DOES_NOT_EXIST;
INVALID_SEARCH_CRITERIA	GLOBAL_SEARCH_SUPPORTS_AT_MOST_THREE_KEYWORDS_AND_REQUIRES_AT_LEAST_ONE_KEYWORDS_ARE_COMPOSED_OF_ONLY_LETTERS_DIGITS_AND_DASHES;
INVALID_SEARCH_FIELD_OBJ	WS_ERROR_INVALID_SEARCH_CUSTOMFIELD_OBJECT;
INVALID_SEARCH_FIELD_OBJ	WS_INVALID_SEARCH_FIELD_OBJECT;
INVALID_SEARCH_MORE	WS_INVALID_SEARCHMORE_OPERATION;
INVALID_SEARCH_OPERATOR	WS_INVALID_SEARCH_OPERATOR;
INVALID_SEARCH_PAGE_INDEX	WS_INVALID_SEARCH_PAGE_INDEX;

Error Code Returned	Long Description or Message
INVALID_SEARCH_PAGE_SIZE	WS_INVALID_SEARCH_PAGE_SIZE;
INVALID_SEARCH_VALUE	WS_MISSING_SEARCH_VALUE;
INVALID_SEARCH_VALUE	WS_MISSING_SEARCH_VALUES;
INVALID_SERIAL_OR_LOT_NUMBER	SERIAL_AND_LOT_NUMBERS_MAY_NOT_CONTAIN_THE_1_CHARACTER;
INVALID_STATE	SIGNUP_PROSPECT_STATE_1_IS_INVALID;
INVALID_SUPERVISOR	YOU_CANT_INSERT_THIS_EMPLOYEE_RECORD_AS_IT_WOULD_CREATE_A_LOOP_IN_THE_SUPERVISOR_HIERARCHY;
INVALID_SUPERVISOR	EMPLOYEES_CAN_NOT_BE_THEIR_OWN_SUPERVISOR;
INVALID_TAX_VALUE	GST_AND_PST_AMOUNT_CANNOT_BE_NEGATIVE;
INVALID_TAX_VALUE	GST_TAX_VALUE_IS_NOT_A_VALID_NUMBER_1;
INVALID_TIME	_1_IS_NOT_A_VALID_TIME
INVALID_TO_DATE	INVALID_TO_DATE;
INVALID_TRACKING_NUM	YOU_HAVE_ENTERED_A_TRACKING_NUMBER_THAT_EXCEEDS_THE_MAXIMUM_SIZE_OF_1_CHARACTERS;
INVALID_TRANS	THIS_TRANSACTION_IS_NOT_VALID;
INVALID_TRANSACTION_DATE	THERE_ARE_NO_ACCOUNTING_PERIODS_THAT_COVER_THIS_TRANSACTION_DATE;
INVALID_TRANSACTION_DATE	TRANSACTION_DATE_1_IS_NOT_VALID;
INVALID_URL	PLEASE_BEGIN_THE_1_URL_WITH_BHTTPBNBSPNBSPOBNBSPNBHTT PSBPEXAMPLES_OF_A_VALID_1URL_AREBRBHTTPWWWMYDOMAINCOM IMAGEGIFBNBSPNBSPOBNBSPNBHTTTPSONETWOORGUSERNAMETEST JPG;
INVALID_VAT_AMOUNT	VAT_AMOUNT_CANNOT_BE_NEGATIVE;
INVALID_WS_VERSION	MESSAGE_CONTAINS_A_WEB_SERVICES_VERSION_THAT_DOES_NOT_EXIST
INVALID_YEAR_FORMAT	ILLEGAL_YEAR_FORMAT_OR_VALUE_EXAMPLES_1999_2000_2001_ETC;
INVALID_ZIP_FILE	INVALID_ARCHIVE_ZIP_FILE_MUST_CONTAIN_AT_LEAST_ONE_FILE;
INVENTORY_NUM_DISALLWD	INVENTORY_NUMBERS_ARE_ONLY_ALLOWED_ON_ITEMS_WITH_SERIAL_NUMBERED_OR_LOT_NUMBERED_ITEMS;
ITEM_ACCT_REQD	ONE_OF_THE_ITEMS_ON_THIS_TRANSACTION_HAS_AN_AMOUNT_BUT_NO_ACCOUNT__PLEASE_FIX_THE_ITEM_AND_RESUBMIT_THE_TRANSACTION;
ITEM_ACCT_REQD	ONE_OF_THE_ITEMS_ON_THIS_TRANSACTION_HAS_AN_AMOUNT_BUT_NO_ACCOUNT__PLEASE_FIX_THE_ITEM_AND_RESUBMIT_THE_TRANSACTION__IT_MIGHT_BE_THAT_YOU_HAVE_RECENTLY_ELECTED_TO_CHARGE_FOR_SHIPPING_AND_HAVE_NOT_ASSIGNED_AN_ACCOUNT_TO_THE_SHIPPING_ITEM_THAT_IS_INCLUDED_IN_THIS_TRANSACTION;
ITEM_ACCT_REQD	YOU_MUST_SPECIFY_ASSET_AND_COGS_ACCOUNTS_FOR_THIS_INVENTORY_ITEM;

Error Code Returned	Long Description or Message
ITEM_COUNT_MISMATCH	COGSCORRECTION_2_MEANS_OF_CALCULATING_THE_ITEM_COUNT_DO_NOT_MATCH_FOR_ITEM_1_VS_2;
ITEM_COUNT_MISMATCH	COGSCORRECTION_2_MEANS_OF_CALCULATING_THE_ITEM_COUNT_DO_NOT_MATCH_FOR_ITEM_1_VS_2_THERE_ARE_TRANSACTIONS_IN_THE_SYSTEM_IN_WHICH_THIS_ITEM_IS_USED_BUT_THE_ASSET_ACCOUNT_FOR_THAT_ITEM_IS_NOT_THE_CURRENT_ASSET_ACCOUNT_IN_THE_ITEM_RECORD;
ITEM_IS_UNAVAILABLE	ITEM_IS_UNAVAILABLE;
ITEM_NAME_MUST_BE_UNIQUE	AN_ITEM_WITH_THAT_NAME_ALREADY_EXISTS_PLEASE_CHOOSE_ANOTHER_NAME;
ITEM_PARAM_REQD_IN_URL	ERROR_ITEM_PARAMETER_IDNNN_WAS_NOT_PROVIDED_ON_THE_URL;
LIST_KEY_REQD	WS_MISSING_LIST_KEY;
LOCATION_REQD	YOU_MUST_SPECIFY_A_LOCATION_IN_ORDER_TO_USE_1_NUMBERS_WHEN_MULTILLOCATION_INVENTORY_IS_ENABLED;
LOCATIONS_IN_USE	YOUR_CLASSES_CANNOT_BE_CONVERTED_TO_LOCATIONS_BECAUSE_YOUR_EXISTING_LOCATION_RECORDS_ARE_REFERRED_TO_BY_TRANSACTIONS_OR_OTHER_RECORDS;
LOCATIONS_SETUP_REQD	YOU_MUST_FIRST_DEFINE_LOCATIONS_LISTSLOCATIONSNEW_BEFORE_YOU_CAN_DISTRIBUTE_INVENTORY;
LOCATIONS_SETUP_REQD	YOU_MUST_FIRST_DEFINE_LOCATIONS_LISTSLOCATIONSNEW_BEFORE_YOU_CAN_TRANSFER_INVENTORY;
LOGIN_DISABLED	LOGIN_ACCESS_HAS_BEEN_DISABLED_FOR_THIS_ROLE;
LOGIN_DISABLED	YOUR_ACCESS_TO_1_HAS_BEEN_DEACTIVATED_PLEASE_CONTACT_THE_COMPANY_S_ADMINISTRATOR_TO_REACTIVATE_YOUR_ACCESS;
LOGIN_DISABLED	YOUR_ACCESS_TO_THIS_ACCOUNT_HAS_BEEN_REMOVED_OR_DISABLED_PLEASE_CONTACT_THE_ACCOUNT_ADMINISTRATOR;
LOGIN_DISABLED	INVALID_LOGIN_CUSTOMER_ACCESS_IS_DISABLED;
LOGIN_DISABLED_PARTNER_CENTER	DISABLED_LOGIN_ADVANCED_PARTNER_CENTER_ACCESS_HAS_BEEN_DISABLED_BY_THE_ACCOUNT_ADMINISTRATOR;
LOGIN_DISABLED_PARTNER_CENTER	DISABLED_LOGIN_STANDARD_PARTNER_CENTER_ACCESS_HAS_BEEN_DISABLED_BY_THE_ACCOUNT_ADMINISTRATOR;
LOGIN_EMAIL_REQD	INVALID_LOGIN_YOU_MUST_PROVIDE_AN_EMAIL_ADDRESS;
LOGIN_NAME_AND_PSWD_REQD	PLEASE_ENTER_BOTH_A_USER_NAME_AND_A_PASSWORD;
MACHN_LIST_KEY_NAMES_REQD	WS_MISSING_MACHINE_LIST_KEY_NAMES;
MANDATORY_PRD_TYPE_REQD	PLEASE_SELECT_THE_MANDATORY_PERIOD_TYPE;
MATRIX_INFO_TEMP_LOST	MATRIX_ITEM_INFORMATION_WAS_LOST_THIS_WAS_PROBABLY_DUE_TO_A_TRANSIENT_CONDITION_LIKE_A_SERVER_REBOOT_PLEASE_TRY_AGAIN;
MERGE_OPERATION_DISALLWD	YOU_CANNOT_PERFORM_MERGE_OPERATIONS_ON_RECORDS_THAT_BE_LONG_TO_YOUR_GROUP;

Error Code Returned	Long Description or Message
EXCEEDED_MAX_FIELD_LENGTH	
MAX_RCRDS_EXCEEDED	WS_EXCEEDED_MAX_RECORDS;
MEDIA_FILE_INVALID_JSCRIPT	MEDIA_FILE_WAS_OF_TYPE_JAVASCRIPT_AND_WOULD_NOT_COMPILE_ERROR_ON_LINE;
MEDIA_NOT_FOUND	MEDIA_ITEM_NOT_FOUND_1;
MEDIA_NOT_INITIALIZED	MEDIA_ITEM_CANNOT_BE_INITIALIZED;
MISMATCHED_SEARCH_PARENT_HESIS	SEARCH_ERROR_PARENTHESES_ARE_UNBALANCED;
MISSING_ACCT_PRD	THERE_ARE_NOT_ENOUGH_PERIODS_SPECIFIED_TO_COMPLETELY_RECOGNIZE_THIS_TRANSACTION;
MISSNG_ACCT_PRD	UNABLE_TO_FIND_AN_ACCOUNTING_PERIOD_FOR_THE_ALLOCATION_DATE;
MISSNG_REV_REC_RCRD	UNABLE_TO_LOCATE_REVENUE_RECOGNITION_RECORDS;
MISSNG_SO_REV_REC_PARAMS	UNABLE_TO_GET_REVENUE_RECOGNITION_PARAMETERS_FROM_ORIGINATING_SALES_ORDER;
MISSNG_SO_START_END_DATES	UNABLE_TO_ACQUIRE_START_AND_END_DATE_FROM_SALES_ORDER;
MLTPLE_TAX_LINES_DISALLWD	MULTIPLE_TAX_LINES_FOR_LINE_ITEM_IN_TRANSACTION;
MORIZATION_JE_ALERT	MORIZATION_JOURNAL_ENTRY_ALERT;
MORIZED_TRAN_CANNOT_DELETE	MORIZED_TRAN_CANNOT_DELETE;
MORIZED_TRAN_FAILURE_REMEDY	MORIZED_TRAN_FAILURE_REMEDY;
MORIZED_TRAN_WITH_SAME_NAME	MORIZED_TRAN_WITH_SAME_NAME;
MSNG_FIELD_OWRTE_MUST_BE_TRUE	MISSINGFIELDWRITE_ATTRIBUTE_REQUIRED_AS_TRUE;
MULTISELECT_TYPE_REQD	WS_NO_MULTISELECT_TYPE_IS_DEFINED;
NAME_ALREADY_IN_USE	A_MASS_UPDATE_HAS_ALREADY_BEEN_SAVED_WITH_THAT_NAME_PLEASE_USE_A_DIFFERENT_NAME;
NAME_ALREADY_IN_USE	A_SEARCH_HAS_ALREADY_BEEN_SAVED_WITH_THAT_NAME_PLEASE_USE_A_DIFFERENT_NAME;
NAME_NOT_AUTHORIZED	RT_NAME_NOT_AUTHORIZED_EXCEPTION;
NAME_TYPE_FLDR_FIELDS_REQD	MISSING_REQUIRED_FIELDS__NAME_TYPE_AND_FOLDER;
NEGATIVE_TAX_RATE_DISALLWD	A_TAX_RATE_CANNOT_BE_NEGATIVE;
NO_DATA_FOUND	NO_DATA_WAS_FOUND;
NO_EXPENSES_FOR_PRD	THE_ALLOCATION_SOURCES_OR_DESTINATIONS_DID_NOT_HAVE_ANY_EXPENSES_ASSOCIATED_WITH_THEM_FOR_THE_SELECTED_PERIOD;
NO_MTRX_ITEMS_TO_UPDATE	THERE_ARE_NO_MATRIX_SUBITEMS_TO_UPDATE;

Error Code Returned	Long Description or Message
NO_ORD_SHPMNT	THERE_IS_NO_SHIPMENT_ON_THAT_ORDER;
NO_RCRD_FOR_USER	THERE_IS_NO_RECORD_FOR_THIS_USER_IN_THE_COMPANYS_ENTITY_TABLE_EMAILLOGINSEMAIL1_KENTITY2;
NO_SCHDUL_APPLIED	THERE_WERE_NO_SCHEDULES_THAT_NEED_TO_APPLIED_TO_THE_GIVEN_PERIOD;
NO_SCHDUL_APPLIED	THERE_WERE_NO_SCHEDULES_THAT_NEED_TO_BE_APPLIED_TO_THE_INPUT_ACCOUNTING_PERIOD;
NONZERO_AMT_REQD	YOU_DID_NOT_ENTER_NONZERO_AMOUNTS_FOR_ANY_ACCOUNTS;
NOT_IN_INVT	YOU_MAY_NOT_DISTRIBUTE_1_NUMBERS_THAT_ARE_NOT_CURRENTLY_IN_INVENTORY__YOU_ATTEMPTED_TO_DISTRIBUTE_THE_FOLLOWING_1_NUMBERS_THAT_WERE_NOT_IN_INVENTORY_2;
NONMATCHING_EMAILS	EMAIL_ADDRESSES_DONT_MATCH;
NULL_CHECK_NUMBER	NULL_CHECK_NUMBER;
NUM_ALREADY_USED	CHECK_NUMBER_X_IS_ALREADY_USED;
NUM_ITEMS_GRTR_THAN_QTY	NUMBER_OF_ITEM_NUMBERS_ENTERED_1_IS_GREATER_THAN_THE_ITEM_QUANTITY_2;
NUM_ITEMS_NOT_EQUAL_TO_QTY	NUMBER_OF_ITEM_NUMBERS_ENTERED_1_IS_NOT_EQUAL_TO_THE_ITEM_QUANTITY_2;
NUM_REQD_FOR_FIRST_LABEL	NO_NUMBER_WAS_SPECIFIED_FOR_THE_FIRST_LABEL;
ONE_ADMIN_REQD_PER_ACCT	THIS_OPERATION_WOULD_LEAVE_YOUR_ACCOUNT_WITHOUT_AN_ACTIVE_ADMINISTRATOR__IN_ORDER_TO_SUCCESSFULLY_PERFORM_THE_MASS_UPDATE_PLEASE_DESELECT_AT_LEAST_ONE_ENTITY_WITH_AN_ADMINISTRATOR_ROLE;
ONE_ADMIN_REQD_PER_ACCT	YOU_CANT_DELETE_THIS_EMPLOYEE_NO_ADMINISTRATORS_FOR_THIS_ACCOUNT_WOULD_REMAIN;
ONE_ADMIN_REQD_PER_ACCT	YOU_CANT_INACTIVATE_1_THE_ACCOUNT_WOULD_BE_LEFT_WITH_NO_ACTIVE_ADMINISTRATORS;
ONE_ADMIN_REQD_PER_ACCT	YOU_CANT_REMOVE_THE_ADMINISTRATOR_ROLE_FROM_THIS_USER_NO_ADMINISTRATORS_FOR_THIS_ACCOUNT_WOULD_REMAIN;
ONE_EMPL_REQD	AT_LEAST_ONE_EMPLOYEE_IS_REQUIRED_TO_PROCESS_PAYROLL;
ONE_RCRD_REQD_FOR_MASS_UPDATE	PLEASE_CREATE_AT_LEAST_ONE_1_BEFORE_USING_THIS_MASS_UPDATE;
ONE_ROLE_REQD	YOU_CANT_INACTIVATE_ALL_ROLES_YOU_WOULD_NOT_BE_ABLE_TO_LOG_IN;
ONLINE_BILL_PAY_SETUP_REQD	B1B_IS_NOT_SET_UP_FOR_ONLINE_BILL_PAY_TO_SET_UP_THIS_PAYEE_CLICK_GO_BACK;
ONLINE_FORM_ID_REQD	MISSING_REQUIRED_ONLINE_FORM_ID;
ONLINE_ORD_FEATURE_DISABLED	CANT_OPEN_STORE_FOR_1__THIS_COMPANY_DOES_NOT_HAVE_THE_BONLINE_ORDERINGB_FEATURE_ENABLED__THE_FEATURE_IS_REQUIRED_FOR_CUSTOMERS_TO_MAKE_ONLINE_PURCHASES;

Error Code Returned	Long Description or Message
ONLINE_ORD_FEATURE_DISABLED	CANT_OPEN_STORE_FOR_1__THIS_COMPANY_DOES_NOT_HAVE_THE_B USE_SALES_ORDERSB_FEATURE_ENABLED__THE_FEATURE_IS_REQUIRE D_FOR_CUSTOMERS_TO_MAKE_ONLINE_PURCHASES;
ONLY_ONE_CONTRIB_ITEM_REQD	ONLY_ONE_INSTANCE_OF_A_COMPANY_CONTRIBUTION_ITEM_IS_ALLO WED_ON_AN_EMPLOYEE_RECORD;
ONLY_ONE_DEDCT_ITEM_REQD	ONLY_ONE_INSTANCE_OF_A_DEDUCTION_ITEM_IS_ALLOWED_ON_AN_ EMPLOYEE_RECORD;
ONLY_ONE_DISTRIB_ALLWD	YOU_MAY_NOT_DISTRIBUTE_1_NUMBERS_MORE_THAN_ONCE__YOU_A TTEMPTED_TO_DISTRIBUTE_THE_FOLLOWING_1_NUMBERS_MORE_THA N_ONCE_2;
ONLY_ONE_EARNING_ITEM_REQD	ONLY_ONE_INSTANCE_OF_AN_EARNING_ITEM_IS_ALLOWED_ON_AN_E MPLOYEE_RECORD;
ONLY_ONE_UNIT_AS_BASE_UNIT	ONLY_ONE_UNIT_MAY_BE_DESIGNATED_AS_THE_BASE_UNIT;
ONLY_ONE_UPLOAD_ALLWD	YOU_CANNOT_UPLOAD_MORE_THAN_ONE_FILE_AT_A_TIME;
ONLY_ONE_WITHLD_ITEM_REQD	ONLY_ONE_INSTANCE_OF_A_WITHHOLDING_ITEM_IS_ALLOWED_ON_A N_EMPLOYEE_RECORD;
ORD_ALREADY_APPRVD	YOU_CANNOT_CANCEL_THIS_ORDER_BECAUSE_IT_HAS_ALREADY_BEEN _APPROVED;
ORDER_DSNT_EXIST	THAT_ORDER_DOES_NOT_EXIST;
PACKAGE_WEIGHT_REQD	ATTEMPTED_TO_CREATE_A_PACKAGE_WITHOUT_SPECIFYING_A_NONZE RO_PACKAGE_WEIGHT;
PARENT_CANT_ITSELF_BE_MEMBER	PARENT_ITEM_CAN_NOT_BE_A_MEMBER_OF_ITSELF;
PARTNER_ACCESS_DENIED	PARTNERS_DO_NOT_HAVE_ACCESS_TO_THIS_ITEM;
PARTNER_CODE_ALREADY_USED	A_PARTNER_WITH_THAT_PARTNER_CODE_ALREADY_EXISTS;
PAYCHECK_IN_USE	YOU_CANNOT_CLEAR_THIS_PAYCHECK_BECAUSE_IT_IS_LINKED_TO_BY _ONE_OR_MORE_LIABILITY_PAYMENTS__YOU_MUST_DELETE_OR_VOID _THOSE_TRANSACTIONS_FIRST;
PAYEE_REQD	U_MUST_SPECIFY_A_PAYEE;
PAYEE_REQD_FOR_PMT	YOUR_PAYMENT_HAS_BEEN_RECORDED_BUT_AN_ONLINE_BILL_PAY_P AYMENT_WILL_NOT_BE_MADE_BECAUSE_NO_PAYEE_WAS_SPECIFIEDY OU_SHOULD_RETURN_TO_THE_PAYMENT_SCREEN_IF_YOU_WISH_TO_P RINT_THE_CHECK;
PAYROLL_EXPENSE_ACCT_REQD	PLEASE_SELECT_A_EXPENSE_ACCOUNT_FOR_PAYROLL_ITEM;
PAYROLL_EXPENSE_ACCT_REQD	PLEASE_SELECT_AN_EXPENSE_ACCOUNT_FOR_PAYROLL_ITEM;
PAYROLL_FEATURE_DISABLED	YOU_HAVE_NOT_ENABLED_THE_PAYROLL_FEATURE;
PAYROLL_FEATURE_UNAVAILABLE	YOU_ARE_TRYING_TO_EDIT_A_PAY_CHEQUE__PAYROLL_IS_NOT_AVAIL ABLE_IN_NETLEDGER_CANADA;

Error Code Returned	Long Description or Message
PAYROLL_ITEM_DELETE_DISALLWD	UNABLE_TO_REMOVE_PAYROLL_ITEM_1__THERE_ARE_EXISTING_TRANSACTIONS_FOR_THIS_PAYROLL_ITEM__YOU_MAY_MARK_IT_INACTIVE_INSTEAD;
PAYROLL_LIABILITY_ACCT_REQD	PLEASE_SELECT_A_LIABILITY_ACCOUNT_FOR_PAYROLL_ITEM;
PERMISSION_VIOLATION	PERMISSION_VIOLATION_YOU_MAY_NO_LONGER_EDIT_THIS_RECORD;
PERMISSION_VIOLATION	PERMISSION_VIOLATION_YOU_MAY_NOT_ACCESS_THIS_RECORD;
PHONE_NUM_REQD	PLEASE_PROVIDE_A_PHONE_NUMBER;
PLAN_IN_USE	THIS_PLAN_HAS_ALREADY_BEEN_USED_TO_GENERATE_COMMISSION_CALCULATIONS_AND_CANT_BE_DELETED;
PMT_ALREADY_SBMTD	THIS_PAYMENT_HAS_ALREADY_BEEN_SUBMITTED_FOR_ONLINE_BILL_PAY;
PMT_EDIT_DISALLWD	THIS_LIABILITY_PAYMENT_CANNOT_BE_EDITED_WHILE_IT_HAS_AN_AUTOMATED_CLEARING_HOUSE_TRANSMISSION_IN_PROCESS;
POSITIVE_QTY_REQD	ASSEMBLY_MEMBER_ITEMS_MUST_HAVE_POSITIVE_QUANTITIES;
POSTING_PRD_SETUP_REQD	CREATION_OF_JOURNAL_ENTRIES_REQUIRE_A_SINGLE_ACCOUNTING_PERIOD_VALUE_ACROSS_ALL_REVENUE_RECOGNITION_EVENTS__PLEASE_SETUP_A_POSTING_PERIOD_FILTER;
PRD_SETUP_REQD	YOU_MUST_CHANGE_YOUR_PERIOD_DEFINITIONS_TO_CONTAIN_FISCAL_YEARS_PLEASE_VISIT_SETUPMANAGE_ACCOUNTING_PERIODS_AND_CLICK_SET_UP_YEAR;
PRD_SETUP_REQD	YOU_MUST_DEFINE_THE_PERIODS_OF_THE_PRIOR_FISCAL_YEAR_PLEASE_VISIT_SETUPMANAGE_ACCOUNTING_PERIODS_AND_CLICK_SET_UP_YEAR;
PREF_VENDOR_COST_REQD	DROP_SHIPSPECIAL_ORDER_ITEMS_MUST_HAVE_A_PREFERRED_VENDOR_AND_A_COST;
PREFERRED_TAX_AGENCY_REQD	A_PREFERRED_TAX_AGENCY_HAS_BEEN_DELETED__PLEASE_CHOOSE_A_NEW_ONE;
PREFERRED_TAX_AGENCY_REQD	ERROR_NO_PREFERRED_TAX_AGENCIES_HAVE_BEEN_SET_UP;
PRDS_DISALLWD_NAMES_NOT_UNIQUE	AFTER_ADDING_NEW_PERIODS_NOT_ALL_NAMES_WOULD_BE_UNIQUE;
PRIVATE_RCRD_ACCESS_DISALLWD	YOU_CANNOT_VIEW_OR_EDIT_THIS_RECORD_BECAUSE_IT_IS_MARKED_PRIVATE;
PRIVATE_STATUS_CHNG_DISALLWD	YOU_CANNOT_MAKE_THIS_CONTACT_PRIVATE;
PSWD_REQD	YOU_MUST_PROVIDE_A_PASSWORD_TO_GIVE_THIS_PERSON_ACCESS_TO_YOUR_ACCOUNT;
PSWD_REQD	PASSWORD_IS_EMPTY;
PSWD_REQD	PLEASE_TYPE_YOUR_PASSWORD_INTO_BOTH_FIELDS;
PWSDS_DONT_MATCH	PASSWORDS_DONT_MATCH;

Error Code Returned	Long Description or Message
PWSDS_DONT_MATCH	THE_PASSWORDS_YOU_ENTERED_DO_NOT_MATCH__PLEASE_REENTER_YOUR_PASSWORDS;
PWSDS_DONT_MATCH	THE_PASSWORDS_YOU_HAVE_ENTERED_DO_NOT_MATCH;
PWSDS_DONT_MATCH	NEW_PASSWORDS_DONT_MATCH;
RCRD_DSNT_EXIST	THAT_RECORD_DOES_NOT_EXIST; Note: Users encounter this error when they attempt to update or delete a record with a non-existent record key.
RCRD_PREVSLY_DELETED	THIS_RECORD_HAS_ALREADY_BEEN_DELETED;
RCRD_PREVSLY_DELETED	THIS_RECORD_HAS_BEEN_DELETED_SINCE_THE_LIST_WAS_GENERATED;
RCRD_TYPE_REQD	WS_MISSING_RECORD_TYPE;
RCRD_UNEDITABLE	THAT_RECORD_IS_NOT_EDITABLE;
READY_MEMORIZED_TRANS	READY_MEMORIZED_TRAN;
RECOGNIZED_METHOD	RECOGNIZED_METHOD;
RECUR_EVENT_DISALLWD	A_YEARLY_EVENT_CANNOT_BE_ON_THE_29TH_OF_FEBRUARY;
REG_CRNCY_NOT_BASE_CRNCY	THE_CURRENCY_YOU_ARE_REGISTERED_TO_USE_IS_DIFFERENT_FROM_THE_BASE_CURRENCY_OF_THIS_COMPANY;
REQUEST_PARAM_REQD	THIS_REQUEST_IS_MISSING_A_REQUIRED_PARAMETER;
REV_REC_DATE_REQD	NO_REVENUE_RECOGNITION_START_DATE_SPECIFIED;
REVERSAL_DATE_WARNING	REVERSAL_DATE_WARNING;
ROLE_REQUIRED	In order to login, a role is required unless a default has been previously set.
ROUNDING_DIFF_TOO_BIG	ROUNDING_DIFFERENCE_TOO_BIG__TAX1_1_TAX2_2;
ROUNDING_ERROR	ROUNDING_ERROR_1;
SCHDUL_EDIT_DISALLWD	THIS_SCHEDULE_CANNOT_BE_EDITED_AS_IT_HAS_ALREADY_BEEN_USED_FOR_COMMISSION_CALCULATIONS_PLEASE_GO_BACK_AND_SELECT_SAVE_AS_NEW_INSTEAD;
SEARCH_INTEGER_REQD	PLEASE_ENTER_AN_INTEGER_NUMBER_TO_SEARCH_ON;
SECTIONS_UNEDITABLE	ILTIN_SECTIONS_CANNOT_BE_EDITED_EXCEPTION;
SECURE_TRANS_REQD_ON_CHECKOUT	STORE_SERVER_ERROR_AS_CONFIGURED_THIS_SERVER_DOES_NOT_PERMIT_SECURE_TRANSACTIONS_REQUIRED_BY_STORE_CHECKOUT;
SESSION_TERMD_2ND_LOGIN_DECTD	SOMEONE_HAS_LOGGED_IN_AS_THIS_USER_FROM_A_DIFFERENT_COMPUTER_OR_BROWSER_WINDOW_ONLY_ONE_PERSON_MAY_LOGIN_AS_A_GIVEN_USER_AT_A_TIME_AS_A_CONSEQUENCE_THIS_SESSION_HAS_BEEN_TERMINATED_PTO_HAVE_MULTIPLE_WINDOWS_LOGGED_IN_AS_THE_SAME_USER_USE_FILENEWWINDOW_FRO
SESSION_TERMD_2ND_LOGIN_DECTD	YOU_CAN_HAVE_A_MAXIMUM_OF_1_ACTIVE_USERS_AT_A_TIME_IN_2;
SESSION_TERMD_2ND_LOGIN_DECTD	YOU_CAN_HAVE_A_MAXIMUM_OF_1_ACTIVE_USERS_AT_A_TIME_THAT_ARE_ENABLED_FOR_OFFLINE_SALES_CLIENT;
SESSION_TIMED_OUT	WS_SESSION_TIMED_OUT;

Error Code Returned	Long Description or Message
SESSION_TIMED_OUT	YOUR_SESSION_HAS_TIMED_OUT_PLEASE_REENTER_YOUR_INFORMATION_AND_TRY_AGAIN;
SESSION_TIMED_OUT	YOUR_CONNECTION_HAS_TIMED_OUT__PLEASE_A_HREFPAGESLOGINJS_P_TARGETSELFLOG_INA_AGAIN;
SO_HAS_CHILD_TRANS	SALESORDER_HAS_CHILD_TRANSACTIONS;
START_DATE_AFTER_END_DATE	THE_START_DATE_MUST_PRECEED_THE_END_DATE;
STORAGE_LIMIT_EXCEEDED	YOU_ENTERED_A_VALUE_THAT_WILL_EXCEED_THE_INTERNAL_STORAGE_LIMIT_OF_1_PLEASE_REDUCE_THE_NUMBER;
STORE_ALIAS_UNAVAILABLE	THE_STORE_ALIAS_YOU_CHOSE_1_IS_ALREADY_TAKEN__PLEASE_GO_BACK_AND_CHOOSE_ANOTHER;
STORE_DOMAIN_UNAVAILABLE	THE_STORE_DOMAIN_NAME_YOU_CHOSE_0_IS_ALREADY_TAKEN_PLEASE_GO_BACK_AND_CHOOSE_ANOTHER;
STORE_SERVER_ERROR	STORE_SERVER_ERROR_AS_CONFIGURED_THIS_SERVER_DOES_NOT_PERMIT_SHOPPING;
SUBITEM_REQD	YOU_MUST_FIRST_SELECT_THE_NEW_SUBITEMS_ON_THE_MATRIX_TAB_YOU_WANT_TO_ADD;
SUBITEM_REQD	YOU_MUST_FIRST_SELECT_THE_SUBITEMS_ON_THE_MATRIX_TAB_YOU_WANT_TO_CREATE;
SUCCESS_TRANS	THE_TRANSACTION_WAS_ENTERED_1_SUCCESSFULLY_2;
TAX_CODES_SETUP_PROBLEM	THE_TAX_CODES_HAVENT_BEEN_SET_PROPERLY;
TAX_CODES_SETUP_REQD	THE_COMPANY_IS_NOT_USABLE_ADMINISTRATOR_HASNT_SET_UP_THE_TAX_CODES;
TAX_CODES_SETUP_REQD	CANT_OPEN_STORE_FOR_1__THIS_COMPANY_DOES_NOT_HAVE_ITS_TAX_CODES_FULLY_SET_UP__THIS_IS_REQUIRED_TO_PROPERLY_CALCULATE_TAXES_ON_INTERNATIONAL_OTHERPROVINCE_AND_SAMEPROVINCE_ORDERS;
TICKET_NOT_LOCATED	THE_TICKET_1_CANNOT_BE_LOCATED_IN_THE_ERROR_DATABASE_IF_THIS_IS_FROM_A_CUSTOMER_LOGGED_CASE_THE_ERROR_MAY_NOT_YET_BE_INSERTED_INTO_THE_SYSTEM;
TRANS_AMTS_UNBALNCD	TRANSACTION_IS_NOT_IN_BALANCE__AMOUNTSTAXESSHIPPING_1_TOTAL_AMOUNT_2;
TRANS_APPLIED_AMTS_UNBALNCD	TRANSACTION_IS_NOT_IN_BALANCE__TOTAL_TO_APPLY_OF_1_DOES_NOT_EQUAL_SUM_OF_APPLIED_2_AND_UNAPPLIED_3;
TRANS_APPLIED_AMTS_UNBALNCD	TRANSACTION_IS_NOT_IN_BALANCE__TOTAL_TO_APPLY_OF_1_DOES NOT_EQUAL_SUM_OF_PAYMENT_2_AND_CREDITS_3;
TRANS_CLASS_UNBALNCD	TRANSACTION_OUT_OF_BALANCE_FOR_CLASS_1_TOTAL_2;
TRANS_DEPT_UNBALNCD	TRANSACTION_OUT_OF_BALANCE_FOR_DEPARTMENT_1_TOTAL_2;
TRANS_DOES_NOT_EXIST	NO_TRANSACTION_EXISTS_FOR_THAT_ENTITY;
TRANS_EDIT_DISALLWD	YOU_CANNOT_EDIT_THIS_TRANSACTION_1_DOES_NOT_SUPPORT_THE_IMPORTED_TRANSACTION;
TRANS_EDIT_DISALLWD	THIS_TRANSACTION_IS_IN_A_PERIOD_THAT_HAS_BEEN_CLOSED_YOU_MAY_NOT_EDIT_IT;

Error Code Returned	Long Description or Message
TRANS_FORGN_CUR_UNBALNCD	TRANSACTION_WAS_NOT_IN_BALANCE_FOREIGN_CURRENCY_POSTING_TOTAL_1;
TRANS_FORGN_CUR_UNBALNCD	TRANSACTION_WAS_NOT_IN_BALANCE_FOREIGN_CURRENCY_TOTAL_1;
TRANS_IN_USE	THIS_TRANSACTION_CANNOT_BE_DELETED_BECAUSE_IT_IS_LINKED_TO_BY_ONE_OR_MORE_TRANSACTIONS;
TRANS_LINE_AND_PMT_UNBALNCD	TRANSACTION_IS_NOT_IN_BALANCE__LINE_ITEM_SUM_OF_1_NOT_EQUAL_TO_PAYMENT_AMOUNT_2;
TRANS_LINES_UNBALNCD	TRANSACTION_IS_NOT_IN_BALANCE__LINE_ITEM_SUM_OF_1_DOES_NOT_EQUAL_AMOUNT_OF_2;
TRANS_LINES_UNBALNCD	TRANSACTION_IS_NOT_IN_BALANCE__LINE_ITEM_SUM_OF_1_DOES_NOT_EQUAL_APPLIED_AMOUNT_OF_2;
TRANS_LOC_UNBALNCD	TRANSACTION_OUT_OF_BALANCE_FOR_LOCATION_1_TOTAL_2;
TRANS_NOT_CLEAVED	TRANSACTION_NOT_CLEAVED_UP;
TRANS_NOT_COMPLETED	TRANSACTION_WAS_NOT_COMPLETE;
TRANS_UNBALNCD	TRANSACTION_IS_NOT_IN_BALANCE_1;
TRANS_UNBALNCD	TRANSACTION_IS_NOT_IN_BALANCE_12_OTHERCOUNT_3;
TRANS_UNBALNCD	TRANSACTION_WAS_NOT_IN_BALANCE_POSTING_TOTAL_1;
TRANS_UNBALNCD	TRANSACTION_WAS_NOT_IN_BALANCE_TOTAL_1;
TRANSACTION_DELETED	THE_TRANSACTION_YOU_ARE_ATTEMPTING_TO_ACCESS_HAS_BEEN_DELETED;
TYPE_NOT_RECGNZD	DIA_TYPE_NOT_RECOGNIZED;
UNDEFINED_ACCTNG_PRD	THE_ACCOUNTING_PERIOD_RANGE_1_HAS_NOT_BEEN_DEFINED_PLEASE_VISIT_A_HREFAPPSETUPPERIODFISCALPERIODSNLSETUP__ACCOUNTING__MANAGE_ACCOUNTING_PERIODSA_TO_DEFINE_THIS_PERIOD_OR_SET_UP_YOUR_YEAR;
UNDEFINED_ACCTNG_PRD	THE_COMPARISON_ACCOUNTING_PERIOD_RANGE_1_HAS_NOT_BEEN_DEFINED_PLEASE_VISIT_A_HREFAPPSETUPPERIODFISCALPERIODSNLSETUP__ACCOUNTING__MANAGE_ACCOUNTING_PERIODSA_TO_DEFINE_THIS_PERIOD_OR_SET_UP_YOUR_YEAR;
UNDEFINED_ACCTNG_PRD	THE_DEFAULT_ACCOUNTING_PERIOD_FOR_THIS_REPORT_HAS_NOT_BEEN_DEFINED_PLEASE_VISIT_A_HREFAPPSETUPPERIODFISCALPERIODSNLSETUP__ACCOUNTING__MANAGE_ACCOUNTING_PERIODSA_TO_DEFINE_THIS_PERIOD_OR_SET_UP_YOUR_YEAR;
UNDEFINED_TAX_PRD	THE_DEFAULT_TAX_PERIOD_FOR_THIS_REPORT_HAS_NOT_BEEN_DEFINED_PLEASE_VISIT_A_HREFAPPSETUPPERIODTAXPERIODSNLSETUP__ACCOUNTING__MANAGE_TAX_PERIODSA_TO_DEFINE_THIS_PERIOD_OR_SET_UP_YOUR_YEAR;
UNEXPECTED_ERROR	AN_UNEXPECTED_ERROR_HAS_OCCURRED__TECHNICAL_SUPPORT_HAS_BEEN_ALERTED_TO_THIS_PROBLEM;
UNEXPECTED_ERROR	WS_AN_UNEXPECTED_ERROR_OCCURRED;
UNEXPECTED_ERROR	WS.DTO.JAVA.CLASS.IS_NOT_DEFINED;

Error Code Returned	Long Description or Message
UNEXPECTED_ERROR	WS_FAILED_TO_INSTANTIATE_CLASS;
UNEXPECTED_ERROR	WS_NO_DTO_CLASS_IS_DEFINED;
UNEXPECTED_ERROR	WS_SCHEMA_CREATION_FAILED_1_DOES_NOT_HAVE_A_MATCHING_XML_SCHEMA_TYPE;
UNEXPECTED_ERROR	WS_SERVER_ERROR_MISSING_DATABASE_ENTRIES_IN_WSRECORDELEMENT_AND_WSNAMESPACE_TABLES;
UNEXPECTED_ERROR	WS_SERVER_ERROR_NO_FORM_REQUEST_CLASS_IS_DEFINED;
UNEXPECTED_ERROR	AN_ERROR_OCCURRED_WHILE_PROCESSING_ITEM_OPTIONS;
UNEXPECTED_ERROR	ACTION_EXCEPTION;
UNEXPECTED_ERROR	ERROR;
UNEXPECTED_ERROR	ERROR_1;
UNEXPECTED_ERROR	PROBLEM_DURING_COMMISSION_CALCULATION;
UNITS_TYP_IN_USE	THIS_UNITS_TYPE_IS_USED_BY_1_2_YOU_MUST_DELETE_THE_2_AND_ALL_ASSOCIATED_TRANSACTIONS_IN_ORDER_TO_DELETE_THIS_UNITS_TYPE;
UNKNWN_ALLOCTN_FREQ_TYP	UNABLE_TO_DETERMINE_ALLOCATION_FREQUENCY_TYPE;
UNKNWN_ALLOCTN_SCHEDUL_FREQ_TYP	UNABLE_TO_DETERMINE_ALLOCATION_SCHEDULE_FREQUENCY_TYPE;
UNRECOGNIZED_METHOD	UNRECOGNIZED_METHOD_1;
UNSUPRTD_DOC_TYP	UPLOAD_UNSUPPORTED_DOC_TYPE_ERROR;
UNSUPPORTED_WS_VERSION	MESSAGE_CONTAINS_AN_UNSUPPORTED_WEB_SERVICES_VERSION
UPDATE_DISALLWD	WS_UPDATE_IS_NOT_ALLOWED;
UPDATE_PRICE_AMT_REQD	PLEASE_SPECIFY_AN_AMOUNT_TO_UPDATE_PRICES;
URL_ID_PARAM_REQD	URL_IS_MISSING_THE_ID_PARAMETER_THE_FILE_COULD_NOT_BE_RETRIEVED;
URL_REQD	YOU_MUST_ENTER_A_URL_FOR_THIS_MEDIA_ITEM;
USER_DISABLED	USER_DISABLED;
VALID_PRD_REQD	INSERT_TRANSACTION_FAILURE_NO_VALID_OPEN_1_PERIOD_FOR_DATE_2;
VALID_PRD_REQD	UPDATE_TRANSACTION_FAILURE_NO_VALID_OPEN_1_PERIOD_FORDATE_2;
VENDOR_TYPE_REQD	NO_VENDOR_TYPE_WAS_SPECIFIED_IF_CREATING_A_TAX_AGENCY_PLEASE_ENSURE_THAT_THE_VENDOR_TYPE_IS_ACTIVE_AND_MARKED_AS_A_TAX_AGENCY;
VOIDING_REVERSAL_DISALLWD	YOU_MAY_NOT_CREATE_A_VOIDING_REVERSAL_FOR_TRANSACTIONS_WITH_INVENTORY_IMPACT;
WS_CONCUR_SESSION_DISALLWD	WS_INVALID_CONCURRENT_SESSION;
WS_FEATURE_REQD	WS_NO_WEB_SERVICES_FEATURE;
WS_LOG_IN_REQD	WS_LOG_IN_FIRST;

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 [“Handling Errors”](#) on page 56.

Task IDs

The following table lists currently exposed TaskIDs that can be referenced in an https POST for use in single login between a Web services application and the NetSuite UI. For more information, refer to “Web Services to UI Single Login” on page 55.

Task ID	Page Label in NetSuite	URL
EDIT_ACCOUNT	New Accounts	/app/accounting/account/account.nl
EDIT_ACTIVITY	New Activity	/app/crm/calendar/activity.nl
EDIT_ALLOCATION	Create Allocation Schedules	/app/accounting/transactions/allocation.nl
EDIT_AMENDW4	Form W-4	/app/common/entity/amendw4.nl
EDIT_BILLINGSCHEDULE	New Billing Schedule	/app/accounting/otherlists/billingschedule.nl
EDIT_CALL	New Phone Calls	/app/crm/calendar/call.nl
EDIT_CAMPAIGN	New Campaigns	/app/crm/marketing/campaign.nl
EDIT_CAMPAIGNEMAIL	Campaign E-mail Address	/app/crm/marketing/campaignemail.nl
EDIT_CASEFIELDRULE	New Set Up Case Rules	/app/crm/support/casefieldrule.nl
EDIT_CASEFORM	New Online Case Forms	/app/crm/support/caseform.nl
EDIT_CASEISSUE	New Case Issue List	/app/crm/support/caseissue.nl
EDIT_CASEORIGIN	New Case Origin Types	/app/crm/support/caseorigin.nl
EDIT_CASEPRIORITY	New Case Priority	/app/crm/support/casepriority.nl
EDIT_CASESTATUS	New Case Status	/app/crm/support/casestatus.nl
EDIT_CASETERRITORY	New Manage Case Territories	/app/crm/support/supportterritory.nl
EDIT_CASETYPE	New Case Types	/app/crm/support/casetype.nl
EDIT_CLASS	New Classes	/app/common/otherlists/classype.nl
EDIT_COLORTHEME	New Color Themes	/app/setup/look/colortheme.nl
EDIT_COMMISSIONSCHEDULE	New Commission Schedules	/app/crm/sales/commissions/commissionschedule.nl
EDIT_COMPANY	New Company	/app/common/entity/company.nl
EDIT_COMPETITOR	Competitor	/app/crm/sales/competitor.nl
EDIT_CONTACT	New Contacts	/app/common/entity/contact.nl
EDIT_CRMGROUP	New Groups	/app/crm/common/crmgroup.nl
EDIT_CRMMESSAGE	New E-mail	/app/crm/common/crmmessage.nl
EDIT_CRMTEMPLATE	New Marketing Templates	/app/crm/common/merge/marketingtemplate.nl
EDIT_CURRENCY	New Currencies	/app/common/multicurrency/currency.nl
EDIT_CUST_	Custom Record Entry	/app/common/custom/custrecordentry.nl
EDIT_CUSTADDRESSFORM	Address Form	/app/common/custom/custaddressform.nl?e=T

Task ID	Page Label in NetSuite	URL
EDIT_CUSTBODYFIELD	New Transaction Body Fields	/app/common/custom/bodycustfield.nl
EDIT_CUSTCENTER	New Center	/app/common/custom/custcenter.nl
EDIT_CUSTCOLUMNFIELD	New Transaction Column Fields	/app/common/custom/columncustfield.nl
EDIT_CUSTEMAILLAYOUT	New Transaction Form HTML Layouts	/app/common/custom/custemmailayout.nl
EDIT_CUSTENTITYFIELD	New Entity Fields	/app/common/custom/entitycustfield.nl
EDIT_CUSTENTRYFORM	New Entry Forms	/app/common/custom/custentryform.nl
EDIT_CUSTEVENTFIELD	New CRM Fields	/app/common/custom/eventcustfield.nl
EDIT_CUSTFORM	New Transaction Forms	/app/common/custom/custform.nl
EDIT_CUSTITEMFIELD	New Item Fields	/app/common/custom/itemcustfield.nl
EDIT_CUSTJOB	New Customers	/app/common/entity/custjob.nl
EDIT_CUSTLAYOUT	New Transaction Form PDF Layouts	/app/common/custom/custlayout.nl
EDIT_CUSTLIST	New Lists	/app/common/custom/custlist.nl
EDIT_CUSTOMERFIELDRULE	New Set Up Sales Rules	/app/crm/sales/customerfieldrule.nl
EDIT_CUSTOMERFORM	New Online Customer Forms	/app/crm/sales/leadform.nl
EDIT_CUSTOMERSTATUS	New Customer Statuses	/app/crm/sales/customerstatus.nl
EDIT_CUSTOMSCRIPT	New Server Script	/app/common/scripting/customscript.nl
EDIT_CUSTOMSCRIPTRECORD	New Server Script Record	/app/common/scripting/customscriptrecord.nl
EDIT_CUSTOTHERFIELD	New Other Field	/app/common/custom/othercustfield.nl
EDIT_CUSTPROFILE	Customer Profile	/app/common/entity/custprofile.nl?category=billing&sc=4
EDIT_CUSTRECORD	New Record Types	/app/common/custom/custrecord.nl
EDIT_CUSTRECORDFIELD	Custom Record Field	/app/common/custom/custreccustfield.nl
EDIT_CUSTRECORDFORM	New Custom Record Form	/app/common/custom/custrecordform.nl
EDIT_CUSTSECTION	New Center Tab	/app/common/custom/custsection.nl
EDIT_CUSTTASKS	Center Links	/app/common/custom/custtasks.nl
EDIT_DEPARTMENT	New Departments	/app/common/otherlists/departments.nl
EDIT_EDITPROFILE	Employee Profile	/app/common/entity/editprofile.nl
EDIT_EMAILTEMPLATE	New E-mail Templates	/app/crm/common/merge/emailtemplate.nl
EDIT_EMPLCATEGORY	New Employee Directory	/app/site/setup/emplcategory.nl
EDIT_EMPLOYEE	New Employees	/app/common/entity/employee.nl
EDIT_EMPLOYEESFA	New Assign Support Reps	/app/common/entity/employeesfa.nl
EDIT_ESCALATIONRULE	New Escalation Rules	/app/crm/support/escalationfieldrule.nl
EDIT_ESCALATIONTERRITORY	New Manage Escalation Assignment	/app/crm/support/escalationterritory.nl
EDIT_EVENT	New Event	/app/crm/calendar/event.nl

Task ID	Page Label in NetSuite	URL
EDIT_EXPCATEGORY	New Expense Categories	/app/accounting/otherlists/expcategory.nl
EDIT_FAXMESSAGE	New Fax Message	/app/crm/common/merge/faxmessage.nl
EDIT_FAXTEMPLATE	New Fax Templates	/app/crm/common/merge/faxtemplate.nl
EDIT_FISCALPERIOD	Setup Fiscal Period	/app/setup/period/fiscalperiod.nl
EDIT_INFOITEM	New Information Items	/app/site/setup/infoitem.nl
EDIT_ISSUE	New Issue	/app/crm/support/issuedb/issue.nl
EDIT_ISSUETAG	New Issue Tag	/app/crm/support/issuedb/issuetag.nl
EDIT_ITEM	New Items	/app/common/item/item.nl
EDIT_ITEMOPTION	New Transaction Item Options	/app/common/custom/itemoption.nl
EDIT_KBCATEGORY	New Knowledge Base	/app/site/setup/kbcategory.nl
EDIT_KPIREPORT	New KPI Report	/app/center/enhanced/kpireportsetup.nl
EDIT_LEAD	New Leads	/app/common/entity/custjob.nl?stage=lead
EDIT_LOCATION	New Locations	/app/common/otherlists/locationtype.nl
EDIT_MAILMERGE	Bulk Merge	/app/crm/common/merge/mailmerge.nl
EDIT_MAILMESSAGE	New Word Message	/app/crm/common/merge/mailmessage.nl
EDIT_MAILTEMPLATE	New Letter Templates	/app/crm/common/merge/mailtemplate.nl
EDIT_MEDIAITEM	New Media Item	/app/common/media/mediaitem.nl
EDIT_MEDIAITEMFOLDER	New File Cabinet	/app/common/media/mediaitemfolder.nl
EDIT_MEMDOC	New Memorized Transactions	/app/accounting/otherlists/memdoc.nl
EDIT_OTHERLIST	New Other Lists	/app/common/otherlists/otherlist.nl
EDIT_OTHERNAME	New Other Names	/app/common/entity/othername.nl
EDIT_PARTNER	New Partners	/app/common/entity/partner.nl
EDIT_PAYROLLITEM	New Payroll Items	/app/common/item/payrollitem.nl
EDIT_PDFMESSAGE	New PDF Message	/app/crm/common/merge/pdfmessage.nl
EDIT_PDFTEMPLATE	New PDF Templates	/app/crm/common/merge/pdftemplate.nl
EDIT_PERIOD	New Manage Accounting Periods	/app/setup/period/fiscalperiod.nl
EDIT_PLANASSIGN	New Commission Plan	/app/crm/sales/commissions/planassign.nl
EDIT_PRESCATEGORY	New Categories	/app/site/setup/prescategory.nl
EDIT_PROSPECT	New Prospects	/app/common/entity/custjob.nl?stage=prospect
EDIT_REFERRALCODE	New Promotion Codes	/app/crm/sales/referralcode.nl
EDIT_RELATEDITEM	New Related Items Group	/app/site/setup/relateditem.nl
EDIT_RESOLVECONFLICTS	Resolve Conflicts	/app/common/entity/conflictresolution.nl
EDIT_RESOURCE	Resource	/app/crm/calendar/resource.nl
EDIT_REVRECSCHEDULE	New Revenue Recognition Template	/app/accounting/otherlists/revrecschedule.nl

Task ID	Page Label in NetSuite	URL
EDIT_ROLE	New Role	/app/setup/role.nl
EDIT_RPTGROUP	New Financial Statement Section	/app/reporting/reportgroup.nl
EDIT_RPTLAYOUT	New Financial Statement Layout	/app/reporting/reportsectionlayout.nl
EDIT_RSSFEED	New RSS Feed	/app/site/hosting/rssfeed.nl
EDIT_SALESTERRITORY	New Manage Sales Territories	/app/crm/sales/salesterritory.nl
EDIT_SAVEDSEARCH	New Saved Search	/app/common/search/search.nl?cu=T&e=F
EDIT_SEARCH	New Search	/app/common/search/search.nl
EDIT_SHIPITEM	New Shipping Items	/app/common/item/shipitem.nl
EDIT_SITEITEMTEMPLAT	Item Template	/app/site/setup/siteitemtemplate.nl
EDIT_SITEMEDIA	Site Media	/app/site/media/sitemedia.nl
EDIT_SOLUTION	New Solutions	/app/crm/support/kb/solution.nl
EDIT_SSCATEGORY	New Publish Saved Search	/app/site/setup/sscategory.nl
EDIT_STOREITEMLISTLAYOUT	New Layouts	/app/site/setup/storeitemlistlayout.nl
EDIT_STORETAB	New Tabs	/app/site/setup/storetab.nl
EDIT_SUPPORTCASE	New Cases	/app/crm/support/supportcase.nl
EDIT_SYSALERT	System Alert	/app/common/otherlists/systemalert.nl
EDIT_TASK	New Tasks	/app/crm/calendar/task.nl
EDIT_TAXACCT	Tax Control Account	/app/setup/taxacct.nl
EDIT_TAXCODE	New Tax Codes	/app/common/item/taxitem.nl?CA=T
EDIT_TAXITEM	New Sales Tax Items	/app/common/item/taxitem.nl
EDIT_TAXITEMVAT	New Tax Codes	/app/common/item/taxitem.nl?vat=T
EDIT_TAXPERIOD	New Manage Tax Periods	/app/setup/period/taxperiod.nl
EDIT_TAXTYPE	Tax Type	/app/setup/taxtype.nl
EDIT_TOPIC	New Topics	/app/crm/support/kb/topic.nl
EDIT_TRAN_BUILD	Build Assemblies	/app/accounting/transactions/build.nl
EDIT_TRAN_CARDCHRG	Use Credit Card	/app/accounting/transactions/cardchrg.nl
EDIT_TRAN_CASHRFND	Refund Cash Sales	/app/accounting/transactions/cashrfnd.nl
EDIT_TRAN_CASHSALE	Enter Cash Sales	/app/accounting/transactions/cashsale.nl
EDIT_TRAN_CHECK	Write Checks	/app/accounting/transactions/check.nl
EDIT_TRAN_COMMISSN	Individual Commission	/app/accounting/transactions/commissn.nl
EDIT_TRAN_CUSTCHRG	Create Statement Charges	/app/accounting/transactions/custchrg.nl
EDIT_TRAN_CUSTCRED	Issue Credit Memos	/app/accounting/transactions/custcred.nl
EDIT_TRAN_CUSTDEP	Record Customer Deposits	/app/accounting/transactions/custdep.nl
EDIT_TRAN_CUSTINVC	Create Invoices	/app/accounting/transactions/custinvc.nl

Task ID	Page Label in NetSuite	URL
EDIT_TRAN_CUSTPYMT	Accept Customer Payments	/app/accounting/transactions/custpymt.nl
EDIT_TRAN_CUSTRFND	Issue Customer Refund	/app/accounting/transactions/custrfnd.nl
EDIT_TRAN_DEPAPPL	Apply Customer Deposits	/app/accounting/transactions/depappl.nl
EDIT_TRAN_DEPOSIT	Make Deposits	/app/accounting/transactions/deposit.nl
EDIT_TRAN_ESTIMATE	Prepare Estimates	/app/accounting/transactions/estimate.nl
EDIT_TRAN_EXPREPT	Enter Expense Reports	/app/accounting/transactions/exprept.nl
EDIT_TRAN_FXREVAL	Revalue Open Currency Balances	/app/accounting/transactions/fxrevalsetup.nl
EDIT_TRAN_INVADJST	Adjust Inventory	/app/accounting/transactions/invadjst.nl
EDIT_TRAN_INVDIRSTR	Distribute Inventory	/app/accounting/transactions/invdistr.nl
EDIT_TRAN_INVTRNFR	Transfer Inventory	/app/accounting/transactions/invtrnfr.nl
EDIT_TRAN_INVVKSHT	Adjust Inventory Worksheet	/app/accounting/transactions/invvksht.nl
EDIT_TRAN_ITEMRCPT	New Item Receipt	/app/accounting/transactions/itemrcpt.nl
EDIT_TRAN_ITEMSHIP	New Item Fulfillment	/app/accounting/transactions/itemship.nl
EDIT_TRAN_JOURNAL	Make Journal Entries	/app/accounting/transactions/journal.nl
EDIT_TRAN_JOURNALIMPORT	Import Journal Entries	/app/accounting/transactions/journal.nl?imp=T
EDIT_TRAN_LIAADJST	Liability Adjustment	/app/accounting/transactions/liaadjst.nl
EDIT_TRAN_LIABPYMT	Pay Payroll Liabilities	/app/accounting/transactions/liabpymt.nl
EDIT_TRAN_OPPRTNTY	Create Opportunities	/app/accounting/transactions/opprtnty.nl
EDIT_TRAN_PAYCHECK	Individual Paychecks	/app/accounting/transactions/paycheck.nl
EDIT_TRAN_PURCHORD	Enter Purchase Orders	/app/accounting/transactions/purchord.nl
EDIT_TRAN_REPLENISHLOC	Replenish Location	/app/accounting/transactions/replenishloc.nl
EDIT_TRAN_RTNAUTH	Issue Return Authorizations	/app/accounting/transactions/rtnauth.nl
EDIT_TRAN_SALESORD	Enter Sales Orders	/app/accounting/transactions/salesord.nl
EDIT_TRAN_TAXLIAB	Write GST Liability	/app/accounting/transactions/vatliab.nl
EDIT_TRAN_TAXPYMT	Pay Sales Tax	/app/accounting/transactions/taxpymt.nl
EDIT_TRAN_TRANSFER	Transfer Funds	/app/accounting/transactions/transfer.nl
EDIT_TRAN_UNBUILD	Unbuild Assemblies	/app/accounting/transactions/unbuild.nl
EDIT_TRAN_VATLIABAU	Pay Tax Liability	/app/accounting/transactions/vatliab.nl
EDIT_TRAN_VENDBILL	Enter Bills	/app/accounting/transactions/vendbill.nl
EDIT_TRAN_VENDCREC	Enter Vendor Credits	/app/accounting/transactions/vendcred.nl
EDIT_TRAN_VENDPYMT	Pay Single Vendor	/app/accounting/transactions/vendpymt.nl
EDIT_TRAN_YTDADJST	Enter Payroll YTD Amounts	/app/accounting/transactions/ytdadjst.nl
EDIT_TRANSACTIONLIST	Paycheck History	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=Paycheck
EDIT_UNITSTYPE	New Units Of Measure	/app/common/units/unitstypel.nl

Task ID	Page Label in NetSuite	URL
EDIT_URLALIAS	New Promotional URL	/app/setup/urlalias.nl
EDIT_VENDOR	New Vendors	/app/common/entity/vendor.nl
INTL_SECTION	New Tabs	/internal/admin/section.nl
INTL_SECTIONS	Tabs	/internal/admin/sections.nl
INTL_TASKLINK	New Tasklinks	/internal/admin/tasklink.nl
INTL_TASKLINKS	Tasklinks	/internal/admin/tasklinks.nl
LIST_ACCOUNT	Accounts	/app/accounting/account/accounts.nl
LIST_ACTIVITY	Activity List	/app/crm/calendar/activitylist.nl
LIST_ALLOCATION	Allocation Schedules	/app/accounting/transactions/allocschedulelist.nl
LIST_APPROVEACH	Approve Direct Deposits	/app/accounting/transactions/approveach.nl
LIST_APPROVEEFT	Approve Electronic Funds Transfers	/app/accounting/transactions/approveeft.nl
LIST_APPROVEVP	Approve Vendor Payment Transfers	/app/accounting/transactions/approvevp.nl
LIST_BILLINGSCHEDULE	Billing Schedules	/app/accounting/otherlists/billingschedules.nl
LIST_BUDGET	Budgets	/app/accounting/transactions/budgetlist.nl
LIST_BULKOP	Mass Updates	/app/common/bulk/bulkops.nl
LIST_CALENDAR	Calendar/Event	/app/crm/calendar/calendar.nl
LIST_CALL	Phone Calls	/app/crm/calendar/calllist.nl
LIST_CAMPAIN	Campaigns	/app/crm/marketing/campaignlist.nl
LIST_CAMPAINCALENDAR	Campaign Calendar	/app/crm/marketing/campaigncalendar.nl
LIST_CAMPAINEMAIL	Campaign E-mail Addresses	/app/crm/marketing/campaignemails.nl
LIST_CASEFIELDRULE	Set Up Case Rules	/app/crm/support/casefieldrules.nl
LIST_CASEFORM	Online Case Forms	/app/crm/support/caseforms.nl
LIST_CASEISSUE	Case Issue List	/app/crm/support/caseissuelist.nl
LIST_CASEORIGIN	Case Origin Types	/app/crm/support/caseoriginlist.nl
LIST_CASEPRIORITY	Case Priorities	/app/crm/support/caseprioritylist.nl
LIST_CASESTATUS	Case Statuses	/app/crm/support/casestatuslist.nl
LIST_CASETERRITORY	Manage Case Territories	/app/crm/common/automation/territorymanager.nl?case=T
LIST_CASETYPE	Case Types	/app/crm/support/casetypelist.nl
LIST_CCTRAN	View Credit Card Transactions	/app/accounting/transactions/cardtrans.nl
LIST_CHART_ACCOUNT	Chart of Accounts	/app/accounting/account/accounts.nl?report=T&code=COA
LIST_CLASS	Classes	/app/common/otherlists/classlist.nl
LIST_COLORTHEME	Color Themes	/app/setup/look/colorthemes.nl
LIST_COMMISSIONSCHEDULE	Commission Schedule	/app/crm/sales/commissions/commissionscheds.nl

Task ID	Page Label in NetSuite	URL
LIST_COMPANY	All Companies	/app/common/entity/companylist.nl
LIST_COMPETITOR	Competitors	/app/crm/sales/competitorlist.nl
LIST_CONTACT	Contacts	/app/common/entity/contactlist.nl
LIST_CRMGROUP	Groups	/app/crm/common/crmgrouplist.nl
LIST_CRMTEMPLATE	Marketing Templates	/app/crm/common/merge/marketingtemplates.nl
LIST_CURRENCY	Currencies	/app/common/multicurrency/currencylist.nl
LIST_CUSTBODYFIELD	Transaction Body Fields	/app/common/custom/bodycustfields.nl
LIST_CUSTCENTER	Centers	/app/common/custom/custcenters.nl
LIST_CUSTCOLUMNFIELD	Transaction Column Fields	/app/common/custom/columncustfields.nl
LIST_CUSTEMAILLAYOUT	Transaction Form HTML Layouts	/app/common/custom/custemaillayouts.nl
LIST_CUSTENTITYFIELD	Entity Fields	/app/common/custom/entitycustfields.nl
LIST_CUSTENTRYFORM	Entry Forms	/app/common/custom/custentryforms.nl
LIST_CUSTEVENTFIELD	CRM Fields	/app/common/custom/eventcustfields.nl
LIST_CUSTFIELDTAB	Subtabs	/app/common/custom/custfieldtabs.nl
LIST_CUSTFORM	Transaction Forms	/app/common/custom/custforms.nl
LIST_CUSTITEMFIELD	Item Fields	/app/common/custom/itemcustfields.nl
LIST_CUSTJOB	Customers	/app/common/entity/custjoblist.nl?Customer_STAGE=CUSTOMER
LIST_CUSTLAYOUT	Transaction Form PDF Layouts	/app/common/custom/custlayouts.nl
LIST_CUSTLIST	Lists	/app/common/custom/custlists.nl
LIST_CUSTOMCODEFILES	Custom Code Scripts	/app/common/media/mediaitemfolders.nl?folder=-15
LIST_CUSTOMERFIELDRULE	Set Up Sales Rules	/app/crm/sales/customerfieldrules.nl
LIST_CUSTOMERFORM	Online Customer Forms	/app/crm/sales/leadforms.nl
LIST_CUSTOMERSTATUS	Customer Statuses	/app/crm/sales/customerstatuslist.nl
LIST_CUSTOMSCRIPT	Server Scripts	/app/common/scripting/customscriptlist.nl
LIST_CUSTOMSCRIPTCALENDAR	Script Schedule	/app/common/scripting/customscriptcalendar.nl
LIST_CUSTOMSCRIPTRECORD	Server Script Records	/app/common/scripting/customscriptrecordlist.nl
LIST_CUSTOTHERFIELD	Other Custom Fields	/app/common/custom/othercustfields.nl
LIST_CUSTRECORD	Record Types	/app/common/custom/custrecords.nl
LIST_CUSTRECORDFORM	Online Custom Record Forms	/app/common/custom/custrecordforms.nl
LIST_CUSTSECTION	Center Tabs	/app/common/custom/custsections.nl
LIST_DEPARTMENT	Departments	/app/common/otherlists/departmentlist.nl
LIST_DUPLICATES	Mass Duplicate Record Merge	/app/common/entity/manageduplicates.nl
LIST_EMAILTEMPLATE	E-mail Templates	/app/crm/common/merge/emailtemplates.nl

Task ID	Page Label in NetSuite	URL
LIST_EMPLCATEGORY	Employee Directory	/app/site/setup/emplcategories.nl
LIST_EMPLOYEE	Employees	/app/common/entity/employeeelist.nl
LIST_ENTITY	All Entities	/app/common/entity/entitylist.nl
LIST_ESCALATIONRULE	Set Up Escalation Rules	/app/crm/support/escalationfieldrules.nl
LIST_ESCALATIONTERRITORY	Manage Escalation Assignment	/app/crm/common/automation/territorymanager.nl?escalation=T
LIST_EVENT	Event List	/app/crm/calendar/eventlist.nl
LIST_EXPCATEGORY	Expense Categories	/app/accounting/otherlists/expcategories.nl
LIST_FAXTEMPLATE	Fax Templates	/app/crm/common/merge/faxtemplates.nl
LIST_FCSITEFOLDER	Web Site Hosting Files	/app/common/media/mediaitemfolders.nl?folder=-100
LIST_FINCHRG	Assess Finance Charges	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=CustInvc&Transaction_FINCHRG=T
LIST_FORECAST	List Sales Rep Forecast	/app/crm/sales/forecastlist.nl?Forecast_ISTEAM=F
LIST_IMAGE	Images	/app/common/media/mediaitemfolders.nl?folder=-4
LIST_INFOITEM	Information Items	/app/site/setup/infoitemlist.nl
LIST_ISSUE	Issues	/app/crm/support/issuedb/issuelist.nl
LIST_ITEM	Items	/app/common/item/itemlist.nl
LIST_ITEMOPTION	Transaction Item Options	/app/common/custom/itemoptions.nl
LIST_KBCATEGORY	Knowledge Base	/app/site/setup/kbcategories.nl
LIST_KPIREPORT	KPI Reports	/app/center/enhanced/kpireports.nl
LIST_LEAD	Leads	/app/common/entity/custjoblist.nl?Customer_STAGE=LEAD
LIST_LOCATION	Locations	/app/common/otherlists/locationlist.nl
LIST_MAILMERGE	Merge History	/app/crm/common/merge/mailmergehistory.nl
LIST_MAILTEMPLATE	Letter Templates	/app/crm/common/merge/mailtemplates.nl
LIST_MEDIAITEM	Media Items	/app/common/media/mediaitems.nl
LIST_MEDIAITEMFOLDER	File Cabinet	/app/common/media/mediaitemfolders.nl
LIST_MEDIAITEMFOLDER_LOG	Job Status	/app/external/xml/upload/uploadlog.nl?displayType=FILECABINET
LIST_MEMDOC	Memorized Transactions	/app/accounting/otherlists/memdoclist.nl
LIST_MGRFORECAST	List Sales Manager Forecast	/app/crm/sales/forecastlist.nl?Forecast_ISTEAM=T
LIST_MYROLES	My Roles	/app/center/myroles.nl
LIST_OTHERLIST	Other Lists	/app/common/otherlists/otherlists.nl
LIST_OTHERNAME	Other Names	/app/common/entity/othernames.nl
LIST_PARTNER	Partners	/app/common/entity/partnerlist.nl
LIST_PAYROLLBATCH	Payroll Batches	/app/payroll/payrollbatchlist.nl

Task ID	Page Label in NetSuite	URL
LIST_PAYROLLITEM	Payroll Items	/app/common/item/payrollitems.nl
LIST_PDFTEMPLATE	PDF Templates	/app/crm/common/merge/pdftemplates.nl
LIST_PERIOD	Manage Accounting Periods	/app/setup/period/fiscalperiods.nl
LIST_PLANASSIGN	Commission Plan	/app/crm/sales/commissions/commissionplans.nl
LIST_PLANASSIGNS	Plan Assignments	/app/crm/sales/commissions/planassigns.nl
LIST_PRESCATEGORY	Categories	/app/site/setup/prescategories.nl?ctype=PRES
LIST_PROSPECT	Prospects	/app/common/entity/custjoblist.nl?Customer_STAGE=PROSPECT
LIST_QUOTA	Quotas	/app/crm/sales/quotalist.nl
LIST_RECENTRECORDS	Recent Records	/app/common/otherlists/recentrecords.nl
LIST_RECVDFILES	Attachments Received	/app/common/media/mediaitemfolders.nl?folder=-10
LIST_REFERRALCODE	Promotion Codes	/app/crm/sales/referralcodes.nl
LIST_RELATEDITEM	Related Items Group	/app/site/setup/relateditems.nl
LIST_RESOURCE	Resources	/app/crm/calendar/resources.nl
LIST_REVRECSCHEDULE	Revenue Recognition Templates	/app/accounting/otherlists/revrectemplates.nl
LIST_ROLE	Manage Roles	/app/setup/allroles.nl
LIST_RSSFEED	RSS Feeds	/app/site/hosting/rssfeeds.nl
LIST_SALESTERRITORIES	Sales Territory List	/app/crm/sales/salesterritorylist.nl
LIST_SALESTERRITORY	Manage Sales Territories	/app/crm/common/automation/territorymanager.nl?sales=T
LIST_SAVEDASHBOARD	Published Dashboards	/app/center/setup/savedashboards.nl
LIST_SAVEDBULKOP	Saved Mass Updates	/app/common/bulk/savedbulkops.nl
LIST_SAVEDSEARCH	Saved Searches	/app/common/search/savedsearches.nl
LIST_SEARCHRESULTS	Search Results	/app/common/search/searchresults.nl
LIST_SENTFILES	Attachments Sent	/app/common/media/mediaitemfolders.nl?folder=-14
LIST_SHIPITEM	Shipping Items	/app/common/item/shipitems.nl
LIST_SHIPPINGMANIFEST	Shipping Manifest	/app/common/shipping/fedex/shippingmanifest.nl
LIST_SITEITEMTEMPLAT	Item Templates	/app/site/setup/siteitemtemplates.nl
LIST_SITETHemes	Site Templates	/app/site/setup/sitetheme.nl
LIST_SMBIMPORT	SMB Import	/app/external/xml/upload/upload.nl
LIST_SOLUTION	Solutions	/app/crm/support/kb/solutions.nl
LIST_SSCATEGORY	Publish Saved Search	/app/site/setup/sscategories.nl
LIST_STATUSACH	View Direct Deposit Status	/app/accounting/transactions/statusach.nl
LIST_STATUSEFT	View Electronic Funds Transfer Status	/app/accounting/transactions/statuseft.nl
LIST_STATUSVP	View Vendor Payment Status	/app/accounting/transactions/statusvp.nl

Task ID	Page Label in NetSuite	URL
LIST_STOREITEMLISTLAYOUT	Layouts	/app/site/setup/storeitemlistlayouts.nl
LIST_STORETAB	Tabs	/app/site/setup/storetabs.nl
LIST_SUPPORTCASE	Cases	/app/crm/support/caselist.nl
LIST_SYSALERT	System Alerts	/app/common/otherlists/systemalerts.nl
LIST_SYSTEMEMAILTEMPLATE	Setup System E-mail Templates	/app/crm/common/merge/systememailtemplates.nl
LIST_TASK	Tasks	/app/crm/calendar/tasklist.nl
LIST_TAXCODE	Tax Codes	/app/common/item/taxitems.nl?CA=T
LIST_TAXITEM	Sales Tax Items	/app/common/item/taxitems.nl
LIST_TAXITEMVAT	Tax Codes	/app/common/item/taxitems.nl?vat=T
LIST_TAXPERIOD	Manage Tax Periods	/app/setup/period/taxperiods.nl
LIST_TEMPLATEFILES	Template Files	/app/common/media/mediainitemfolders.nl?folder=-9
LIST_TOPIC	Topics	/app/crm/support/kb/topics.nl
LIST_TRAN_BUILD	Assembly Builds	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=Build
LIST_TRAN_CARDCHRG	Credit Card Charges	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=CardChrg
LIST_TRAN_CASHRFND	Cash Refunds	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=CashRfnd
LIST_TRAN_CASHSALE	Cash Sales	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=CashSale
LIST_TRAN_CHECK	Checks	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=Check
LIST_TRAN_COMMISSN	Individual Commission	/app/accounting/transactions/commissns.nl
LIST_TRAN_CUSTCHRG	Statement Charges	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=CustChrg
LIST_TRAN_CUSTCRED	Credit Memos	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=CustCred
LIST_TRAN_CUSTDEP	Customer Deposits	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=CustDep
LIST_TRAN_CUSTINVC	Invoices	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=CustInvc
LIST_TRAN_CUSTPYMT	Customer Payments	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=CustPymt
LIST_TRAN_CUSTRFND	Customer Refunds	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=CustRfnd
LIST_TRAN_DEPAPPL	Deposit Applications	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=DepAppl
LIST_TRAN_DEPOSIT	Deposits	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=Deposit

Task ID	Page Label in NetSuite	URL
LIST_TRAN_ESTIMATE	Estimates	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=Estimate
LIST_TRAN_EXPREPT	Expense Reports	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=ExpRept
LIST_TRAN_FXREVAL	Currency Revaluations	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=FxReval
LIST_TRAN_INVADJST	Inventory Adjustments	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=InvAdjst
LIST_TRAN_INVDISTR	Inventory Distributions	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=InvDistr
LIST_TRAN_INVTRNFR	Inventory Transfers	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=InvTrnfr
LIST_TRAN_INVWKSHT	Inventory Worksheets	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=InvWksht
LIST_TRAN_ITEMRCPT	Item Receipts	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=ItemRcpt
LIST_TRAN_ITEMSHIP	Item Fulfillments	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=ItemShip
LIST_TRAN_JOURNAL	Journal Entries	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=Journal
LIST_TRAN_LIABPYMT	Liability Payments	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=LiabPymt
LIST_TRAN_OPPRTNTY	Opportunities	/app/accounting/transactions/opprrntylist.nl
LIST_TRAN_PAYCHECK	Paychecks	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=Paycheck
LIST_TRAN_PURCHORD	Purchase Orders	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=PurchOrd
LIST_TRAN_REORDER	Items Ordered	/app/common/item/itemsordered.nl
LIST_TRAN_RTNAUTH	Return Authorizations	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=RtnAuth
LIST_TRAN_SALESORD	Sales Orders	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=SalesOrd
LIST_TRAN_TAXLIAB	Tax Liabilities	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=TaxLiab
LIST_TRAN_TAXPYMT	Tax Payments	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=TaxPymt
LIST_TRAN_TRANSFER	Bank Transfers	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=Transfer
LIST_TRAN_UNBUILD	Assembly Unbuilds	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=Unbuild
LIST_TRAN_VATLIABAU	GST Liabilities	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=VatLiab

Task ID	Page Label in NetSuite	URL
LIST_TRAN_VATLIABUK	VAT Liabilities	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=VatLiab
LIST_TRAN_VENDBILL	Bills	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=VendBill
LIST_TRAN_VENDCRED	Bill Credits	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=VendCred
LIST_TRAN_VENDPYMT	Bill Payments	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=VendPymt
LIST_TRAN_YTDADJST	YTD Adjustments	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=YtdAdjst
LIST_TRANSACTION	Transactions	/app/accounting/transactions/transactionlist.nl?Transaction_TYPE=@ALL@
LIST_UNCATSITEITEM	View Uncategorized Items	/app/setup/uncatsiteitems.nl
LIST_UNITSTYPE	Units Of Measure	/app/common/units/unitstypelist.nl
LIST_UPSELLWIZARD	Upsell Manager	/app/crm/sales/upsell/upsellmanager.nl
LIST_URLALIASES	Promotional URLs	/app/setup/urlaliases.nl
LIST_USER	Manage Users	/app/setup/listusers.nl
LIST_VENDOR	Vendors	/app/common/entity/vendorlist.nl
SRCH_ACTIVITY	Search Activities	/app/common/search/search.nl?searchtype=Activity
SRCH_BUDGET	Search Set Up Budget	/app/common/search/search.nl?searchtype=Budget
SRCH_CALENDAR	Search Calendar/Even	/app/common/search/search.nl?searchtype=Calendar
SRCH_CALL	Search Calls	/app/common/search/search.nl?searchtype=Call
SRCH_CAMPAIGN	Search Campaign	/app/common/search/search.nl?searchtype=Campaign
SRCH_CASE	Search Case	/app/common/search/search.nl?searchtype=Case
SRCH_CLASS	Search Class	/app/common/search/search.nl?searchtype=Class
SRCH_COMPANY	Search All Companie	/app/common/search/search.nl?searchtype=Company
SRCH_COMPETITOR	Search Competitors	/app/common/search/search.nl?searchtype=Competitor
SRCH_CONTACT	Search Contact	/app/common/search/search.nl?searchtype=Contact
SRCH_CUSTOMER	Search Customer	/app/common/search/search.nl?searchtype=Customer
SRCH_DEPARTMENT	Search Department	/app/common/search/search.nl?searchtype=Department
SRCH_DOCUMENT	Search File Cabinet	/app/common/search/search.nl?searchtype=Document
SRCH_EMPLOYEE	Search Employee	/app/common/search/search.nl?searchtype=Employee
SRCH_FIRSTVISIT	Search First Site Visit	/app/common/search/search.nl?searchtype=FirstVisit
SRCH_INFOITEM	Search Information Items	/app/common/search/search.nl?searchtype=Infotem
SRCH_ISSUE	Search Issue	/app/common/search/search.nl?searchtype=Issue
SRCH_ITEM	Search Item	/app/common/search/search.nl?searchtype=Item
SRCH_JOB	Search Job	/app/common/search/search.nl?searchtype=Job

Task ID	Page Label in NetSuite	URL
SRCH_LEAD	Search Leads	/app/common/search/search.nl?searchtype=Customer&Customer_STAGE=LEAD
SRCH_LOCATION	Search Location	/app/common/search/search.nl?searchtype=Location
SRCH_MEDIAITEM	Search Media Items	/app/common/search/search.nl?searchtype=Document
SRCH_MEMDOC	Search Memorized Transactions	/app/common/search/search.nl?searchtype=MemDoc
SRCH_PARTNER	Search Partner	/app/common/search/search.nl?searchtype=Partner
SRCH_PRESCATEGORY	Search Categories	/app/common/search/search.nl?searchtype=PresCategory
SRCH_PROMOTION	Search Promotion Code	/app/common/search/search.nl?searchtype=Promotion
SRCH_PROSPECT	Search Prospects	/app/common/search/search.nl?searchtype=Customer&Customer_STAGE=PROSPECT
SRCH_QUOTA	Search Establish Quota	/app/common/search/search.nl?searchtype=Quota
SRCH_REVREC	Search Revenue Recognition Schedules	/app/common/search/search.nl?searchtype=RevRec
SRCH_SALESTERRITORIES	Sales Territory Search	/app/common/search/search.nl?searchtype=SalesTerritory
SRCH_SHOPPINGCART	Search Shopping Cart	/app/common/search/search.nl?searchtype=ShoppingCart
SRCH_SOLUTION	Search Solution	/app/common/search/search.nl?searchtype=Solution
SRCH_TASK	Search Tasks	/app/common/search/search.nl?searchtype=Task
SRCH_TIME	Search Track Time	/app/common/search/search.nl?searchtype=Time
SRCH_TOPIC	Search Topic	/app/common/search/search.nl?searchtype=Topic
SRCH_TRAN_OPPRTNTY	Search Opportunities	/app/common/search/search.nl?searchtype=Opprtnty
SRCH_TRANSACTION	Search Transactions	/app/common/search/search.nl?searchtype=Transaction
SRCH_VENDOR	Search Vendor	/app/common/search/search.nl?searchtype=Vendor
SUPT_CENTER_ROLE	NetSuite Support Center	/app/login/dashboard.nl?id=
TRAN_ADDCONTENT	Add Content	/app/center/setup/addcontent.nl
TRAN_ADDSHORTCUT	Short Cuts	/core/pages/addShortcut.nl
TRAN_APPROVAL_EXPREPT	Approve Expense Reports	/app/accounting/transactions/approval.nl?type=exprept&label=Expense%20Report
TRAN_APPROVAL_PURCHORD	Approve Purchase Requests	/app/accounting/transactions/approval.nl?type=purchord&label=Purchase%20Request
TRAN_APPROVALS	Approvals	/app/accounting/transactions/approvals.nl
TRAN_APPROVECOMMISSN	Approve Commissions	/app/accounting/transactions/approvecommissn.nl
TRAN_AUDIT	View Audit Trail	/app/accounting/transactions/audit.nl
TRAN_BANKRECON	Find Matching Transactions	/app/accounting/transactions/bankrecon.nl
TRAN_BANKVIEW	Online Bank Statement	/app/accounting/transactions/bankview.nl
TRAN_BAS	Business Activity Statement	/app/accounting/reports/intl/bas.nl
TRAN_BILLINGS	Invoice Schedule	/app/accounting/transactions/billings.nl

Task ID	Page Label in NetSuite	URL
TRAN_BILLPAY_LOG	Job Status	/app/external/xml/upload/uploadlog.nl?displayType=BILLPAY
TRAN_BUDGET	Set Up Budgets	/app/accounting/transactions/budgets.nl
TRAN_BULKBILL_LOG	Job Status	/app/external/xml/upload/uploadlog.nl?displayType=BULKBILL
TRAN_BULKFULFILL_LOG	Job Status	/app/external/xml/upload/uploadlog.nl?displayType=BULKFULFILL
TRAN_CALENDARPREFERENCE	Calendar Preferences	/app/crm/calendar/calendarpreference.nl
TRAN_CAMPAIGNSETUP	Set Up Marketing	/app/setup/campaignsetup.nl
TRAN_CONFIRM	Transaction Confirmation	/core/pages/confirm.nl
TRAN_COPY_BUDGET	Copy Budgets	/app/accounting/transactions/copybudget.nl
TRAN_DOMAINS	Set Up Domains	/app/setup/domains.nl
TRAN_EMAILPWD	Change E-mail Password	/app/center/emailpwd.nl
TRAN_EMPLOYEESFA	Assign Reps	/app/common/entity/employeesfa.nl
TRAN_FINCHRG	Assess Finance Charges	/app/accounting/transactions/finchrg.nl
TRAN_FORECAST	Edit Sales Rep Forecast	/app/crm/sales/forecast.nl
TRAN_GENERATEFISCALPERIODS	Generate Fiscal Periods	/app/setup/period/generatefiscalperiods.nl
TRAN_GENERATETAXPERIODS	Generate Tax Periods	/app/setup/period/generatetaxperiods.nl
TRAN_GSTREFUND	Process GST Refund	/app/accounting/transactions/gstrefund.nl
TRAN_IMPACT	GL Impact	/app/accounting/transactions/impact.nl
TRAN_INVOICECUSTOMERS	Invoice Billable Customers	/app/accounting/transactions/invoicecustomers.nl
TRAN_ITEMSHIPPACK	Mark Orders Packed	/app/accounting/transactions/itemshipmanager.nl?type=pack
TRAN_ITEMSHIPSHIP	Mark Orders Shipped	/app/accounting/transactions/itemshipmanager.nl?type=ship
TRAN_JOURNALAPPROVAL	Approve Journal Entries	/app/accounting/transactions/journalapproval.nl
TRAN_LOGINAUDIT	Login Audit	/app/setup/loginAudit.nl
TRAN_MANAGEPAYROLL	Manage Payroll	/app/payroll/managepayroll.nl
TRAN_MGRFORECAST	Edit Sales Manager Forecast	/app/crm/sales/mgrforecast.nl
TRAN_NLVENDOR	Vendor Center	/app/center/nlvendor.nl.nl
TRAN_OPENBAL	Enter Opening Balances	/app/accounting/transactions/openbal.nl
TRAN_ORDERITEMS	Order Items	/app/accounting/transactions/orderitems.nl
TRAN_PAYMENTS	Payments	/app/accounting/transactions/payments.nl
TRAN_PAYROLLRUN	Process Payroll	/app/accounting/transactions/payroll/payrollrun.nl
TRAN_PDF_F940	Annual Federal Unemployment (940)	/app/accounting/transactions/payroll/pdf/pdfframe.nl?type=f940

Task ID	Page Label in NetSuite	URL
TRAN_PDF_F941	Quarterly Federal Tax Return (941)	/app/accounting/transactions/payroll/pdf/pdfframe.nl?type=f941
TRAN_PREVIEWW2	Form W-2 Preview	/app/accounting/reports/w2preview.nl
TRAN_PRINT	Print Checks and Forms	/app/accounting/print/print.nl
TRAN_PRINT_CASHSALE	Print Receipts	/app/accounting/print/printframe.nl?trantype=cashsale&printtype=transaction&method=print
TRAN_PRINT_CHECK	Print Checks	/app/accounting/print/printframe.nl?trantype=check&printtype=transaction&method=print
TRAN_PRINT_COMMERCIALINVOICE	Print Commerical Invoice	/app/accounting/print/commericalinvoice.nl
TRAN_PRINT_CUSTCRED	Print Credits Memos	/app/accounting/print/printframe.nl?trantype=custcred&printtype=transaction&method=print
TRAN_PRINT_CUSTINVC	Print Invoices	/app/accounting/print/printframe.nl?trantype=custinvc&printtype=transaction&method=print
TRAN_PRINT_ESTIMATE	Print Estimates	/app/accounting/print/printframe.nl?trantype=estimate&printtype=transaction&method=print
TRAN_PRINT_INTEGRATEDSHIPPINGLABEL	Print Intetgrated Shipping Labels	/app/accounting/print/printlabels.nl?printtype=integratedshippinglabel&method=print&title=Integrated Shipping Labels
TRAN_PRINT_PACKINGSLIP	Print Packing Slips	/app/accounting/print/printframe.nl?trantype=&printtype=packingslip&method=print
TRAN_PRINT_PAYCHECK	Print Checks	/app/accounting/print/printframe.nl?trantype=paycheck&printtype=transaction&method=print
TRAN_PRINT_PICKINGTICKET	Print Picking Tickets	/app/accounting/print/printframe.nl?trantype=salesord&printtype=pickingticket&method=print
TRAN_PRINT_PURCHORD	Print Purchase Orders	/app/accounting/print/printframe.nl?trantype=purchord&printtype=transaction&method=print
TRAN_PRINT_RTNAUTH	Return Authorizations	/app/accounting/print/printform.nl?printtype=transaction&trantype=rtnauth&method=print&title=Return%20Authorizations
TRAN_PRINT_SALESORD	Print Sales Orders	/app/accounting/print/printframe.nl?trantype=salesord&printtype=transaction&method=print
TRAN_PRINT_SHIPPINGLABEL	Print Shipping Labels	/app/accounting/print/printframe.nl?trantype=&printtype=shippinglabel&method=print

Task ID	Page Label in NetSuite	URL
TRAN_PRINT_STATEMENT	Generate Statements	/app/accounting/print/printframe.nl?trantype=&printtype=statement&method=print
TRAN_PRINT1099	Form 1099-MISC	/app/accounting/reports/NLPrint1099s.nl?mode=frame
TRAN_PRINTBARCODES	Generate Barcodes	/app/accounting/print/printbarcodes.nl?printtype=null&method=print
TRAN_PRINTSTATEMENT	Individual Statement	/app/accounting/print/printstatement.nl
TRAN_PRINTW2	Form W-2	/app/accounting/reports/NLPrintW2s.nl?mode=frame
TRAN_PRINTW3	Form W-3	/app/accounting/reports/NLPrintW3s.nl?mode=frame
TRAN_PROCESSCOMMISSN	Authorize Commissions	/app/accounting/transactions/processcommissn.nl
TRAN_PROCESSORDER	Process Individual Order	/app/accounting/transactions/processorder.nl
TRAN_PURCHORDPROC	Bill Purchase Orders	/app/accounting/transactions/purchordermanager.nl?type=proc
TRAN_PURCHORDRECEIVE	Receive Purchase Order	/app/accounting/transactions/purchordermanager.nl?type=receive
TRAN_QUOTA	Establish Quotas	/app/crm/sales/quota.nl
TRAN_REALLOCITEMS	Reallocate Items	/app/accounting/transactions/reallocitems.nl
TRAN_RECONCILE	Reconcile Bank Statement	/app/accounting/transactions/reconcile.nl
TRAN_RECONCILE_CC	Reconcile Credit Card Statement	/app/accounting/transactions/reconcile.nl?page_type=cc
TRAN_REIMBURSEMENTS	Reimbursements	/app/accounting/transactions/reimbursements.nl
TRAN_REMINDERS	Setup Reminders	/app/center/setup/reminders.nl
TRAN_REVRECCREATEJE	Revenue Recognition Schedules	/app/accounting/transactions/revreccreateje.nl
TRAN_RTNAUTHAPPRV	Approve Return Authorizations	/app/accounting/transactions/returnauthmanager.nl?type=apprv
TRAN_RTNAUTHCREDIT	Refund Returns	/app/accounting/transactions/returnauthmanager.nl?type=credit
TRAN_RTNAUTHRECEIVE	Receive Returned Order	/app/accounting/transactions/returnauthmanager.nl?type=receive
TRAN_SALESORDAPPRV	Approve Sales Orders	/app/accounting/transactions/salesordermanager.nl?type=apprv
TRAN_SALESORDFULFILL	Fulfill Sales Orders	/app/accounting/transactions/salesordermanager.nl?type=fulfill
TRAN_SALESORDPROC	Bill Sales Orders	/app/accounting/transactions/salesordermanager.nl?type=proc
TRAN_SEARCH	Search	/app/common/search/search.nl
TRAN_SHORTCUTS	Add Shortcuts	/app/center/shortcuts.nl
TRAN_SNAPSHOTCOMPOSER	Custom Snapshot Report	/app/reporting/snapshotcomposer.nl
TRAN_SNAPSHOTS	Setup Snapshots	/app/center/setup/snapshots.nl

Task ID	Page Label in NetSuite	URL
TRAN_TAXPERIODS	Generate Tax Reporting Periods	/app/setup/period/generatetaxperiods.nl
TRAN_TIMEAPPROVAL	Approve Time	/app/accounting/transactions/timeapproval.nl
TRAN_TIMEBILL	Track Time	/app/accounting/transactions/timebill.nl
TRAN_TIMEBILL_WEEKLY	Weekly Time Sheet	/app/accounting/transactions/timebill.nl?weekly=T
TRAN_TIMECALC	Calculate Time	/core/pages/timecalc.nl
TRAN_TIMER	Timer	/core/pages/timer.nl
TRAN_USERPREFS	Set Preferences	/app/center/userprefs.nl
TRAN_VAT100	VAT 100	/app/accounting/reports/intl/vat100.nl
TRAN_VENDBILLPURCHORD	New Purchase Order	/app/accounting/transactions/vendbillpurchord.nl
TRAN_VENDPYMTS	Pay Bills	/app/accounting/transactions/vendpymts.nl