should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

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

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

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

Sample Code

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

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

No Excessive Use of the Service

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

Beta Features

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

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

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

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

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

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

- **Authentication Overview** ................................................................. 1
- **Mandatory Two-Factor Authentication (2FA) for NetSuite Access** ............................................. 3
- **Administrators: Review Roles NetSuite Designates as Mandatory 2FA** ........................................... 3
- **FAQ: Updates for Mandatory 2FA** ............................................................ 5
- **Password Requirements and Policies in NetSuite** ........................................................................... 9
  - **NetSuite Password Requirements** ...................................................................................... 9
  - **Password Settings That Can Be Modified** ........................................................................... 9
  - **System-Defined Password Requirements** ........................................................................... 11
  - **PCI Compliance Password Requirements** ........................................................................... 12
  - **User Access Reset Tool** .................................................................................................. 12
  - **Password Reset Tips for Administrators** ........................................................................... 13
- **Session Management in NetSuite** .................................................................................................. 15
  - **Types of NetSuite Sessions** .......................................................................................... 15
  - **User Interface (UI) Sessions** .......................................................................................... 15
  - **Simultaneous Access to More Than One NetSuite Account Type** ............................................. 16
  - **The Offline Notification in the UI** .................................................................................... 17
- **NetSuite Login Pages** ............................................................................................ 18
  - **Types of Login Pages for Your NetSuite Account** ............................................................... 18
  - **Creating Custom Pages for Login to Your NetSuite Account** ................................................. 18
  - **Customizing Login and Logout Behavior** ........................................................................... 20
  - **NetSuite Login Pages and iframe Prohibition** ..................................................................... 21
- **Enabling and Creating IP Address Rules** ..................................................................................... 23
  - **Enable the IP Address Rules Feature** ................................................................................ 24
  - **Create Company IP Address Rules** .................................................................................. 24
  - **Create Individual IP Address Rules** .................................................................................. 25
  - **Create Roles without IP Address Restrictions** ..................................................................... 26
  - **Review or Search for Access Restrictions** ........................................................................... 27
- **Token-based Authentication (TBA)** ....................................................................................... 28
  - **Token-based Authentication (TBA) Tasks for Administrators** .................................................. 28
    - **Getting Started with Token-based Authentication** ........................................................... 29
    - **Managing TBA Tokens in the NetSuite UI** ....................................................................... 34
  - **Token-based Authentication (TBA) for Integration Application Developers** ................. 38
    - **The Three-Step TBA Authorization Flow** ....................................................................... 38
    - **The IssueToken Endpoint** ............................................................................................... 40
  - **Troubleshoot Token-based Authentication (TBA)** .................................................................. 42
    - **TBA and the Login Audit Trail** ....................................................................................... 43
    - **The Signature** .............................................................................................................. 45
    - **The Authorization Headers** ............................................................................................ 47
    - **The RESTlet Base String** .............................................................................................. 48
  - **Token-based Authentication and RESTlets** .................................................................... 51
  - **Token-based Authentication and Web Services** .................................................................. 51
- **Two-Factor Authentication (2FA)** .................................................................................. 53
  - **Managing Two-Factor Authentication** ................................................................................ 54
  - **Designate Two-Factor Authentication Roles** ........................................................................ 56
  - **Users and Trusted Devices for Two-Factor Authentication** ............................................... 57
  - **2FA in the NetSuite Application** ..................................................................................... 58
    - **Reset a User's 2FA Settings** ......................................................................................... 59
  - **Supported Countries: SMS and Voice Call** ......................................................................... 60
- **Device ID Authentication** ........................................................................................................ 72
  - **Device ID and the SCIS SuiteApp** ................................................................................... 72
  - **Managing Devices on the List of devices Page** ................................................................. 73
  - **The Device Record** ........................................................................................................ 74
  - **Creating Device Records Manually** .................................................................................... 75
Authentication Overview

NetSuite supports many types of authentication, for authenticating in the NetSuite User Interface (UI) as well as various authentication methods for API access to NetSuite. In this section, see:
- Authentication in the NetSuite UI
- Authentication for API Access to NetSuite
- Authentication Matrix

Authentication in the NetSuite UI

Familiar to many users is authentication by user credentials, that is, entering an email address and a password to log in to the NetSuite UI. See the help topic Your User Credentials for information for users. Topics for account administrators include Password Requirements and Policies in NetSuite, NetSuite Login Pages, and Enabling and Creating IP Address Rules.

Two-Factor Authentication, or 2FA, can protect your company from unauthorized access to your data. NetSuite offers a free 2FA solution that provides both online and offline methods for receiving verification codes.

**Important:** For enhanced security, NetSuite requires two-factor authentication (2FA) for all Administrator and other highly privileged roles in all NetSuite accounts. This requirement includes access to production, sandbox, development, and Release Preview accounts.

The Administrator and other highly privileged roles are designated as 2FA authentication required by default, and this requirement cannot be removed. Any standard or customized roles that include highly privileged permissions are indicated in the Mandatory 2FA column on the Two-Factor Authentication Roles page.

For more information, see the following topics:
- Mandatory Two-Factor Authentication (2FA) for NetSuite Access
- Mandatory 2FA, the IssueToken Endpoint, and nlauth_otp
- Two-Factor Authentication (2FA)
- Permissions Requiring Two-Factor Authentication (2FA)
- Designate Two-Factor Authentication Roles

The following 2FA videos are also available:
- [2FA Delivered Your Way for Administrators](#)
- [2FA Delivered Your Way for Users](#)

Single Sign-on (SSO) Overview

NetSuite supports several different types of single sign-on (SSO). SSO is a transparent authentication scheme that enables the seamless linking of applications and at the same time maintaining application-specific access control. SSO eliminates the need for users to log in to each application separately.

NetSuite supports the following methods for inbound SSO access to the NetSuite UI:
- See SAML Single Sign-on for inbound SSO using authentication from a third party identity provider compliant with SAML v2.0. See also SAML Single Sign-on Access to Web Store.
- See OpenID Single Sign-on for inbound SSO from Google Apps, using Google OpenID Connect authentication.
Authentication for API Access to NetSuite

NetSuite offers Token-based Authentication, or TBA, enabling client applications to use a token to access NetSuite through APIs. TBA eliminates the need for RESTlets and web services integrations to store user credentials. TBA allows integrations to comply with any authentication policy that is deployed in a NetSuite account for UI login, such as SAML Single Sign-on, Inbound Single Sign-on, and Two-Factor Authentication (2FA). You can use 2FA roles and roles with SAML Single Sign-on permissions with TBA. Although user credentials can be used for both web services integrations and for RESTlets, TBA is a better option.

To understand the effect that the mandatory 2FA requirement for highly privileged roles has on SOAP web services integrations and RESTlets, see the following topics:

- Mandatory Two-Factor Authentication (2FA) for NetSuite Access
- FAQ: Updates for Mandatory 2FA
- Token-based Authentication (TBA)
- Mandatory 2FA, the IssueToken Endpoint, and nlauth_otp
- Integration Management
- Authentication for SOAP Web Services
- Authentication for RESTlets

**Important:** Inbound Single Sign-on was another authentication option for SOAP web services integrations used in the past. However, as of 2018.1, no new solutions using Inbound SSO for SOAP web services are permitted. See the help topic Inbound Single Sign-on. See also the Inbound Single SignOn (Inbound SSO) section in Authentication for SOAP Web Services.

Outbound Single Sign-on, called SuiteSignOn in NetSuite, is another authentication method supported for integrations. See Outbound Single Sign-on (SuiteSignOn). Only SOAP web services is supported for SuiteSignOn calls.

Device ID authentication is also available in NetSuite. Device ID authentication was developed for use with the SuiteCommerce InStore (SCIS) application. However, you could develop your own applications to take advantage of the availability of Device ID authentication in NetSuite. See Device ID Authentication. For more information about the SCIS application, see the help topic SuiteCommerce InStore.

Authentication Matrix

The following table shows the authentication methods supported in NetSuite.

<table>
<thead>
<tr>
<th></th>
<th>NetSuite Application</th>
<th>SuiteCommerce</th>
<th>SOAP web services</th>
<th>SuiteScript RESTlets</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User Credentials</strong></td>
<td>Supported</td>
<td>Supported</td>
<td>Supported, with the exception of 2FA-required roles.</td>
<td>Supported, with the exception of 2FA-required roles.</td>
</tr>
<tr>
<td><strong>Token-based Authentication (TBA)</strong></td>
<td></td>
<td></td>
<td>Supported. You should use TBA for</td>
<td>Supported. You should use</td>
</tr>
</tbody>
</table>
Mandatory Two-Factor Authentication (2FA) for NetSuite Access

For enhanced security, NetSuite requires two-factor authentication (2FA) for all Administrator and other highly privileged roles when logging to any NetSuite account. This requirement includes UI access to production, sandbox, development, and Release Preview accounts. The Administrator and highly privileged roles are designated as 2FA authentication required by default, and this requirement cannot be removed. Certain highly privileged permissions also mandate that a role be 2FA required by default. Any standard or customized roles that include these permissions are indicated in the Mandatory 2FA column on the Two-Factor Authentication Roles page. For more information about highly privileged roles, see the help topic Permissions Requiring Two-Factor Authentication (2FA).

The mandatory 2FA requirement also applies to all non-UI access. Non–UI access means access NetSuite through an Application Programming Interface, or API. Web services and RESTlets are two examples of non–UI access to NetSuite. 2FA-required roles employing user credentials for API authentication will fail.

For more information, see the following topics:
- Administrators: Review Roles NetSuite Designates as Mandatory 2FA
- FAQ: Updates for Mandatory 2FA

Administrators: Review Roles NetSuite Designates as Mandatory 2FA

We encourage Administrators to review mandatory 2FA roles in your NetSuite accounts. If you have not explicitly configured a mandatory 2FA role, the values displayed on the Two-Factor Authentication Roles page are Not required and Per session. As of April 2019, the default value for Duration of Trusted Device has changed from 14 days to 30 days.

Changing the Duration of Trusted Device Value

If you wish to change the default value of the Duration of Trusted Device for a mandatory 2FA role, perform the following procedure.
To change the Duration of Trusted Device value for mandatory 2FA roles:


2. Evaluate each role that NetSuite has marked as Mandatory 2FA Required (see the check marks for in the Two-Factor Authentication Required columns) and determine if 30 days is an acceptable value for Duration of Trusted Device.

3. If the default value is not acceptable:
   a. For that role, change the value in the Two-Factor Authentication Required column from Not required to 2FA authentication required.
   b. Change the Duration of Trusted Device value as desired. If you do not update the value, it will be set to Per session after you click Submit in the last step of this procedure.

   **Note:** Until you change the value of Two-Factor Authentication Required from Not Required to 2FA authentication required, you cannot change the value for Duration of Trusted Device. When you change the value to 2FA authentication required, the Duration of Trusted Device value defaults to Per session. Ensure you update the value for Duration of Trusted Device to reflect your desired value.

4. After reviewing all mandatory 2FA required roles and making any necessary changes, including the associated Duration of Trusted Device values, click Submit.

**Mandatory 2FA Roles and Login to the NetSuite UI**

For any mandatory 2FA roles that are not explicitly configured as described in Changing the Duration of Trusted Device Value procedure, users should anticipate the following behavior. When logging in with one of these roles, users see a box with the text Trust this device for 30 days... on the Logging in... page.

If the Duration of Trusted Device value for mandatory 2FA role has been explicitly configured for any value other than Per session, the text on the Logging in... page reflects the configured value. The user
Administrators: Review Roles NetSuite Designates as Mandatory 2FA

can check the box, and the device will be trusted for the stated duration. For more information, see Users and Trusted Devices for Two-Factor Authentication.

FAQ: Updates for Mandatory 2FA

See the following for more information about mandatory 2FA in your NetSuite account.

■ Q:
■ Q:
■ Q:
■ Q:
■ Q:
■ Q:
■ Q:
■ Q:
■ Q:
■ Q:
■ Q:

What could be affected in my account by the mandatory 2FA requirement?

Integrations and RESTlets that employ user credentials to access NetSuite might be affected by this change, particularly in the following situations:

■ NLAuth for RESTlets. See Using User Credentials for RESTlet Authentication for more information.
■ SOAP web services integrations that use the login operation. See the help topic login for more information.
■ SOAP web services integrations that use the passport element sent as Request Level Credentials (RLC). See the help topic Request-Level Credentials for more information.
■ Inbound SSO integrations that use the mapSso operation. See the help topic mapSso for more information.

Important: In your integrations, you might need to use certain functions that require a highly privileged role. We recommend that you transition these integrations to use token-based authentication (TBA) rather than user credentials, or specify a less-privileged role that does not require 2FA. For more information, see Token-based Authentication (TBA) and Mandatory 2FA, the IssueToken Endpoint, and nlauth_otp. For information about using TBA with your integrations, see the help topic Integration Management.

Why do I need to update integrations and RESTlets?

You must make changes if you are using roles that require two-factor authentication (2FA) and employ user credentials with your RESTlets (NLAuth) or in your SOAP web services integrations.

As of 2018.2, 2FA is mandatory for Administrator and other roles with highly privileged permission for access to the NetSuite UI in all existing NetSuite accounts. The mandatory 2FA requirement also applies
to API authentication (non-UI access) to NetSuite. 2FA-required roles employing user credentials for API authentication will fail.

**Transition to Token-based Authentication (TBA)**

- Transition your RESTlets that use NLAuth to TBA. TBA uses OAuth instead of NLAuth. See Authentication for RESTlets and Required Data for Using TBA with RESTlets.
- Transition your SOAP web services integrations that employ the `login` operation, and those that use the passport element sent as Request Level Credentials (RLC) to use TBA. For information about using TBA with your integrations, see the help topics Integration Management and Mandatory 2FA, the IssueToken Endpoint, and nlauth_otp.

**What kind of updates do I need to make?**

You should begin updating your RESTlet and SOAP web services integrations as soon as possible. You have two alternatives for your RESTlet and SOAP web services integrations:

- Change your integrations so that they do not use highly privileged roles by following the best practice “principle of least privilege”. Avoid using the Administrator role or any of the other highly privileged roles listed in Permissions Requiring Two-Factor Authentication (2FA). For more information on constructing roles, see the help topic Customizing or Creating NetSuite Roles.
- If an integration must use a highly privileged role, change the authentication method from user credentials to token-based authentication (TBA). See the help topic Updating an Integration to Send Token-based Authentication Details. See also Using TBA for RESTlet Authentication (OAuth) and Mandatory 2FA, the IssueToken Endpoint, and nlauth_otp. For general information about the TBA feature, see Token-based Authentication (TBA).

**How do I update my SOAP web services integrations and RESTlets?**

You must update the third-party application and the related integration record in NetSuite. Suggestions for updating your integrations and RESTlets follow:

- **Modify Roles: avoid using highly privileged roles that require 2FA**
  - See the help topic Permissions Requiring Two-Factor Authentication (2FA) for a list of highly privileged roles in NetSuite. If you are using an Administrator role, or any other highly-privileged role, create a new role that has only the permissions required to complete the task. Ensure that the new role does not require 2FA.
  - Customize a standard NetSuite role, and remove unnecessary privileges from that customized role. Ensure that the customized role does not require 2FA.
- **Transition to Token-based Authentication (TBA)**
  - If you use RESTlets: change your integration to use TBA. TBA uses OAuth instead of NLAuth. See Authentication for RESTlets and Required Data for Using TBA with RESTlets.
  - If you use SOAP web services: change your integration to use TBA. For information about using TBA with your integrations, see the help topics Integration Management and Mandatory 2FA, the IssueToken Endpoint, and nlauth_otp.

**Do you have an integration using a highly privileged role that is calling the issuetoken endpoint?**

Your integrations that are calling the issue token endpoint using highly privileged roles and employing user credentials as the authentication method can be affected by the Mandatory 2FA requirement. See the help topics Issue Token and Revoke Token REST Services for Token-Based Authentication, Mandatory 2FA, the IssueToken Endpoint, and nlauth_otp, and Using User Credentials for RESTlet Authentication for more information.
How do I change my Inbound SSO solution?

If you are using the NetSuite version of Inbound Single Sign-on (SSO), it is probable that you use the mapSso operation to create mappings. The mapSso operation employs user credentials. The Administrator role is required to create the initial mapping, so that all other users can create their own mappings for Inbound SSO access. For more information, see the help topic Creating the Initial Mapping of the Administrator Role for Inbound Single Sign-on.

The mandatory 2FA requirement will affect API (that is, non-UI) access to NetSuite. You should start updating your Inbound SSO now. 2FA authentication required roles employing user credentials for API authentication will fail.

All existing Inbound SSO mappings will continue to work. However, if you need to create a new mapping for a 2FA required role, you have several alternatives.

- A preferable solution is to switch from Inbound SSO to Token-based Authentication (TBA). We urge you to consider using less-privileged roles along with TBA.
- If you must create the initial mapping of an Administrator role in a new account:
  - Create the mapping manually. Do not use the mapSso operation, it will not work when mandatory 2FA is enabled.
  - Or:
    - Make the mapping part of the application, redirecting a user to NetSuite with a token. This option requires some development effort. For more information, see the help topic Creating the Initial Mapping of the Administrator Role for Inbound Single Sign-on.
- For creating mappings for all other (non-administrator) roles:
  - Consider using less privileged roles that do not require 2FA, and use the mapSso operation. The mapSso operation will work for mapping these less-privileged roles.
  - If a non-privileged role is not an option, and Inbound SSO is still desired, follow the instructions for creating the mapping for an administrator role above (create the mapping manually, or make the mapping part of the application.)

What is wrong with user credentials in my integrations?

You should use tokens for authentication (token-based authentication, or TBA) instead of user credentials (a username and password) with SOAP web services and RESTlets. Why? Passwords expire, but tokens do not, which makes tokens better than passwords for computer-to-computer communications. Update your integrations to use Token-based Authentication (TBA) instead of user credentials.

I do not own the code for my integration, I cannot modify it. What should I do?

It is possible that you might not be able to modify an integration, for example, if your integration was provided by a partner or a third-party provider. If this is the case, contact the partner or third-party who provided the integration, and request that they make the appropriate changes.

What is a highly privileged role?

Highly privileged roles are the Administrator or other highly privileged role, or any of the roles listed in Permissions Requiring Two-Factor Authentication (2FA). Avoid using these roles with your integrations or RESTlets.

What is the principle of least privilege?

When you create a web service (an integration), you associate a role with that integration. Sometimes the task requires a highly privileged role, sometimes not. The principle of least privilege means you should
use a role with the lowest possible privilege that will get the job done. Do not use Administrator role or any other highly privileged role for an integration unless it is absolutely necessary.

**Is SuiteAnalytics Connect access (ODBC or JDBC or ADO.NET) affected by mandatory 2FA?**
No. SuiteAnalytics Connect access is not subject to the mandatory 2FA requirement.

**Are Suitelets or Scheduled Scripts set to have Administrator on the "Execute As Role" field affected?**
No. The mandatory 2FA requirement is applied only to the authentication process.
Password Requirements and Policies in NetSuite

For information about password requirements and policies, see the following:

- NetSuite Password Requirements
- Password Settings That Can Be Modified
- System-Defined Password Requirements
- PCI Compliance Password Requirements
- Password Reset Tips for Administrators

NetSuite Password Requirements

For NetSuite users who log in with a non-customer center role, password validation is based on a combination of the following:

- Account settings that can be modified by administrators. See Password Settings That Can Be Modified.
- System requirements that cannot be modified. See System-Defined Password Requirements.
- PCI DSS requirements that apply to users with the View Unencrypted Credit Cards permission. See PCI Compliance Password Requirements.

Note: Users are locked out for 30 minutes after six consecutive attempts to log in to NetSuite with an incorrect password. For more information, see Password Reset Tips for Administrators.

Password Settings That Can Be Modified

Password settings can be modified by account administrators at Setup > Company > General Preferences. See the following for more information:

- Password Policy
- Minimum Password Length
- Password Expiration in Days

Password Policy

Built-in password policies support three levels of password validation for NetSuite users. These policies enforce the following requirements for password length and content:

- **Strong**: minimum length of 10 characters, at least three of these four character types — uppercase letters, lowercase letters, numbers, non-alphanumeric ASCII characters
- **Medium**: minimum length of eight characters, at least two of these four character types — uppercase letters, lowercase letters, numbers, non-alphanumeric ASCII characters
- **Weak (Not Recommended)**: minimum length of six characters

Note the following details about password policies:
The selected password policy determines the minimum acceptable value for the Minimum Password Length field. The policy does not affect the Password Expiration in Days field value.

All NetSuite accounts are set to a Strong policy by default.

**Important:** The Strong password policy was set as the default for each account in 2014.1. The Strong policy is enforced for all new users added after 2014.1. However, this policy is only enforced for users who existed before the upgrade when these users change their passwords. It is strongly recommended that you set a value for Password Expiration in Days to ensure that the Strong policy is applied to users that existed before the upgrade. (See Password Expiration in Days for more information.) You also can enable the Require Password Change on Next Login option on employee records. You can use CSV import to update this option on many employee records at the same time.

It is possible to reset the password policy to Medium or Weak, but this is not recommended.

**Warning:** If any users in your account have the View Unencrypted Credit Cards permission, PCI password requirements take precedence. See PCI Compliance Password Requirements for more information.

If a user has access to multiple NetSuite accounts that have different password policies, the strongest policy is enforced for that user. A user is defined as an email and password pairing.

The password policy is not applied to users logging in to NetSuite with a customer center role and to customers who register on your website. See Customer Roles and Passwords for more information.

### Minimum Password Length

The Minimum Password Length is the minimum number of characters required for user passwords. Note the following details:

- The default value for this field is determined by the selected password policy. Because the default password policy is Strong, the default Minimum Password Length is 10 characters.
- You can make the minimum password length value longer than the minimum required by the policy. You cannot make this value shorter.
- Minimum password length for customer center roles is six characters. See Customer Roles and Passwords for more information.

### Password Expiration in Days

The Password Expiration in Days is the number of days a login password can be used before a user is prompted to change it.

- Days are calculated from the date that each user last changed their password, not from the date that the company preference is changed.

**Note:** As of December 2015, valid values are 1-365. Values entered before that date are not affected by this limit. However, if any data on the General Preferences page is changed, only valid values within this range will be accepted for the Password Expiration in Days field. For accounts provisioned after this date, the value for Password Expiration in Days is set to 180 days by default.

As of 2013.2 or later, a value of 180 days is the default for all new accounts, ensuring password rotation at least every six months. The value of the Password Expiration in Days field was not reset.
for accounts that existed before 2013.2. Administrators of these accounts should set this value to a minimum of 180 days.

- To comply with Payment Card Industry (PCI) standards, employees with access to view unencrypted credit card numbers are automatically required to change their passwords every 90 days, unless the limit set here is shorter. See PCI Compliance Password Requirements for more information.
- Dates of the previous password change and current password expiration are displayed in the user’s My login audit portlet.

For information about Customer Center roles, see Customer Roles and Passwords.

System-Defined Password Requirements

The following password requirements are always enforced by the system and cannot be changed by account administrators:

- A prior password cannot be reused.
- There must be a significant difference between a new password and the last password. (For example, a user cannot change a password from MyWord!123 to MyWord!145.)
- Easy-to-guess passwords, such as common names, words, and strings like abc123456, are prohibited.
- Non-ASCII characters are considered illegal characters and are prohibited.
- The minimum password length must be at least the minimum required by the selected password policy.
- Passwords must contain the appropriate variety of character types specified by the selected password policy:
  - Character types are:
    - Uppercase alphabet (A, B, ... Z)
    - Lowercase alphabet (a, b, ... z)
    - Number (1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
    - Non-alphanumeric ASCII characters, for example ` ~ ! @ # $ % ^ & * ) ; ' [ ] { } .

Immediate Feedback on Password Changes

As they enter a new password, users receive immediate feedback on compliance with password requirements. You receive the same kind of feedback when you enter a user password on the Access tab of an employee, partner, vendor, or customer record.

For more information about how users can change their passwords, see the help topic Change Password Link.
Note: The Password Criteria fields are shown on any page where a user changes a password. It ensures that the user can tell whether the proposed password meets the security rules enforced by the system.

**Change Password**

Current Password *

New Password *

Confirm New Password *

**Password Criteria**

- Does not contain illegal characters
- Is at least 10 characters long
- Is sufficiently different from previous password
- Contains at least 3 of these 4 character types:
  - Uppercase alpha characters (A, B, ..., Z)
  - Lowercase alpha characters (a, b, ..., z)
  - Numbers (1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
  - Non-alphanumeric ASCII characters (@#$%^&*+=()-_)
- New passwords match

**PCI Compliance Password Requirements**

When using the Credit Card Payments feature, be aware of the Payment Card Industry Data Security Standard (PCI DSS) password requirements. Users with the View Unencrypted Credit Cards permission must change their NetSuite passwords at least every ninety (90) days.

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 users with the View Unencrypted Credit Cards permission.

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 greater requirement remains in effect.

All users with access to unencrypted credit card numbers must change passwords to comply with the PCI requirements.

**User Access Reset Tool**

There are self-service actions users can take when they forget their password, need to update their security questions, or change their own phone number for two-factor authentication (2FA). The User Access Reset page provides administrators with one place to assist users who need help with:

- resetting a NetSuite password
- clearing security questions
- unlocking access to NetSuite
- resetting (2FA) settings

To reset a user's NetSuite access:

2. On the User Access Reset page, enter the email address of the user who requires your help.
3. Check the appropriate box or boxes. You can check multiple boxes if the user needs help with more than one thing.
   - **Initiate Password Reset**: check this box to send an email to the user containing a link so that the user can reset their own password.
   - **Clear Users Security Questions**: check this box to clear the user’s security questions. The user will be prompted to set up new security questions and answers after the next login to NetSuite.
   - **Unlock The User's Access**: check this box to unlock NetSuite access for a user who is locked out of NetSuite after submitting six consecutive incorrect passwords.
   - **Reset 2FA Settings**: check this box to reset (or clear) the user’s settings for 2FA. The user will be prompted to enter new 2FA settings after the next login to NetSuite with a 2FA required role.
4. Click **Save**.

**Password Reset Tips for Administrators**

Users are locked out for 30 minutes after six consecutive attempts to log in to NetSuite with an incorrect password. In most cases, changing a NetSuite password is self-service. However, there are occasions when an administrator must change a user’s password. This can happen, for example, when users forget the answers to their own security questions.

Employee, partner, and vendor roles are considered non-customer center roles. Customers have customer center roles. One person could use the same email address (the NetSuite username) and could be assigned both non-customer center roles and a customer center role. However, these would be treated by the system as two different users, because the information is maintained separately. Changing the password for non-customer center roles has no effect on the password of the customer center role.

**Password Reset for Employees, Partners, and Vendors**

There are several methods for resetting an employee, partner, or vendor password.

- **Self-service password reset**: On the NetSuite login page, a user can click the **Forgot Your Password?** link. The user will receive an email with a link to reset the password. The link in the email will expire after 60 minutes. See the help topic **Getting Access When You Forget Your Password** for information for users.
- **Administrator-initiated password reset:**

  ![Important]

  An administrator must have access to all of the accounts to which a user has access to change that user's password.

  - The **User Access Reset Tool** lets administrators assist users who are not able to reset a password, update security questions, or change their phone number for two factor authentication. You can also reset a user who is locked out of NetSuite after submitting six consecutive incorrect passwords.
  - You should use the **User Access Reset Tool**, but you can initiate a password reset on an employee, partner, or vendor entity record. See the help topic **Changing a User's NetSuite Password**.

**Customer Roles and Passwords**

There are two ways to create customers:
When an administrator (or any user with the necessary permission) creates a Customer record in NetSuite and assigns a user a customer center role.

When a visitor to your website registers an email address and creates a password. This action creates Lead record in your NetSuite account. Lead and Prospect records can be converted into Customer records.

Password Reset for Customers

There are several methods for resetting a customer's password.

- **Self-service password reset:**
  - If the Customer Access feature is enabled in this account, a user can click the **Forgot Your Password?** link on the NetSuite Customer Center login page. The user will receive an email with a link to reset the password. The link in the email will expire after 60 minutes. See Types of Login Pages for Your NetSuite Account and Creating Custom Pages for Login to Your NetSuite Account for more information.
  - On a website login page, customers can click the **Forgot Your Password?** (or similarly named) link. The customer will receive an email with a link to reset the password. The link in the email will expire after 60 minutes. For more information, see the help topic Web Store Password Recovery Email Messages.

- **Administrator-initiated password reset:** This is similar to the initial password setup when the Customer record was created. For instructions, see the help topic Changing a User's NetSuite Password.

Password Requirements for Customers

- The minimum password length for customers is six characters. Passwords in existence for customers before 2014.1 were not affected by this change.
- NetSuite password policies and requirements (other than the requirement for a minimum password length of six characters) do not apply to customers.
- The value set in the account for the Password Expiration in Days field is not applied to customer passwords.
- Setting up security questions and answers is not required for customers.
Session Management in NetSuite

In accordance with industry-wide security recommendations, idle session timeout, absolute session timeout, and session rotation policies are in effect in NetSuite accounts. This section also contains information about managing NetSuite UI sessions and managing sessions when accessing different types of NetSuite accounts.

See the following sections for more information:

- Types of NetSuite Sessions
- User Interface (UI) Sessions
- Simultaneous Access to More than One NetSuite Account Type
- The Offline Notification in the UI

Types of NetSuite Sessions

There are different types of NetSuite sessions. Each type of session is managed independently from the others.

<table>
<thead>
<tr>
<th>Type of Session</th>
<th>Timeout Values</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Interface (UI)</td>
<td>■ Idle session timeout: default is 180 minutes</td>
<td>See User Interface (UI) Sessions for more information about UI session management and timeout values.</td>
</tr>
<tr>
<td></td>
<td>■ Absolute session timeout: 12 hours</td>
<td></td>
</tr>
<tr>
<td>Web services</td>
<td>■ Idle session timeout: 20 minutes</td>
<td>Your integrations should use sessionless protocols based on request level credentials, such as Token-based Authentication (TBA). See Token-based Authentication (TBA) for more information. See also Authentication for SOAP Web Services.</td>
</tr>
<tr>
<td></td>
<td>■ Absolute session timeout: 60 minutes</td>
<td>If your SOAP web services integrations use sessions, you must ensure that your SOAP calls are able to handle session timeouts and reconnection. For more information, see the help topic Session Management for SOAP Web Services.</td>
</tr>
<tr>
<td></td>
<td>■ Operation session timeout: 15 minutes.</td>
<td>If an operation takes more than 15 minutes to complete, consider using asynchronous calls to complete the operation.</td>
</tr>
<tr>
<td>SuiteCommerce</td>
<td>■ Idle session timeout: 20 minutes</td>
<td>After 20 minutes of inactivity in a NetSuite-hosted web store, the user is logged out and becomes an anonymous shopper. There is no automatic relogin to the web store. For more information, see the help topic Web Store Sessions.</td>
</tr>
</tbody>
</table>

User Interface (UI) Sessions

The following timeout values are in effect for the NetSuite UI:

- By default, the idle session timeout value is 180 minutes (3 hours). An administrator can configure the Idle Session Timeout in Minutes value for an account on the General Preferences page. Go to Setup > Company > Preferences > General Preferences. Valid values range from 15 minutes to 720 minutes (12 hours).
- For users logged in with a role that has permission to view unencrypted credit card data, idle session timeout occurs after 15 minutes of inactivity. This restriction is in compliance with section 8.1.8 of
the Payment Card Industry Data Security Standard (PCI DSS) Requirements and Security Assessment Procedures, version 3.2. Click here to view a PDF of this document from the PCI library.

- The default value of 12 hours for absolute session timeout is aligned with the National Institute of Standards and Technology (NIST) Digital Identity Guidelines for Authentication and Lifecycle Management. Click here to view Section 4.2.3, Reauthentication, in the NIST guidelines.

UI session management information for users:

- Users are shown a warning with a 60-second countdown before an idle session timeout occurs. The user can click a **Keep Session Active** button to resume the session.
- Session management across multiple tabs has been synchronized. When a user logs in to an account, all open tabs associated with that account are simultaneously unlocked. When a user logs out of an account, all open tabs associated with that account are locked.
- For users who often switch between roles or different companies and leave multiple browser tabs open from previous sessions, the tabs of stale sessions are shown as inactive. When a user changes roles, sessions from previous roles are invalidated, and those browser tabs are locked.
- Occasionally, users might notice **Office** near the bottom right of the UI. For more information, see The Offline Notification in the UI.

Simultaneous Access to More than One NetSuite Account Type

Most users log in to their production account (**system.netsuite.com**) to perform tasks in NetSuite. Some users may also want simultaneous access to another NetSuite account type, for example, a sandbox account or a Release Preview account. In the past, accessing both account types required using separate browsers to avoid invalidating one session or the other. One benefit of account-specific domains in the NetSuite UI is that accessing two or more account types at the same time is more straightforward.

**To access more than one account type at the same time:**

This procedure describes accessing a sandbox account, but also applies to accessing other account types (Release Preview or development accounts, for example).

1. Open your browser and log in to your NetSuite production account.
2. From the Change Role list, select a role in your sandbox account, right-click, and select open in a different tab.
Simultaneous Access to More than One NetSuite Account Type

3. Click the first tab, the tab with your production account role.
4. Click Login and log in to NetSuite again.

Now both accounts have active sessions: one tab with your production account role, and one tab with your sandbox role.

The Offline Notification in the UI

Occasionally, users might notice near the bottom right of the UI. This Offline notification could indicate a failure in your connection to NetSuite due to network connectivity issues or problems with page performance. The Offline notification warns of a potential problem, but does not necessarily indicate a connectivity failure. The Offline notification can also indicate that a browser page or tab is unresponsive.

The following list contains things you can try to determine the source of the problem.

- Open a new tab in your browser and attempt to open another website. If you cannot access another website, check your ethernet or wireless connection. Contact your account administrator or network administrator if you cannot access any websites.
- If your connection to the internet appears to be working, in a new tab in your browser, open another page in NetSuite. If that is successful, return to the tab where you were working in NetSuite. Save your work. If the save is successful, continue working in NetSuite.
- If you are not able to save your work in NetSuite, but other websites are working well, there might be a problem with NetSuite.
  - Ask your coworkers if they also see the Offline popup in NetSuite.
  - Go to the NetSuite Status page at https://status.netsuite.com to see if there have been any problems reported.
- Contact your account administrator or network administrator if you continue to experience problems with your connection to NetSuite. They might need to investigate your company network, or create a case with Customer Support. Take note of the specific NetSuite pages or forms you are using when you see the Offline notice appear. Also note the tasks you are performing when you see the Offline notification.
NetSuite Login Pages

See the following for more information about login pages for your NetSuite account:

- Types of Login Pages for Your NetSuite Account
- Creating Custom Pages for Login to Your NetSuite Account
- Customizing Login and Logout Behavior
- NetSuite Login Pages and iFrame Prohibition

Types of Login Pages for Your NetSuite Account

There are two different types of login pages for access to the NetSuite UI.

- **Standard login pages**: The standard login pages are `https://system.netsuite.com/pages/customerlogin.jsp` or `https://system.netsuite.com/pages/login.jsp`. These login pages are for users with any role except for a user logging in with a Customer Center role.

- **Customer Center login pages**: We also provide a unique login page for each NetSuite account for your users logging in with a Customer Center role. The URLs for this type of login page are in the following format: `system.<DC>.netsuite.com/app/login/secure/privatelogin.nl?c=<ACCOUNTID>`, where `<DC>` is the identifier of the data center where the account is hosted, and `<ACCOUNTID>` is the unique ID of your account. Administrators can go to Setup > Company > Setup Tasks > Company Information to view the Account ID field and Customer Center Login field. The Customer Center Login field displays the unique URL for the system-provided Customer Center login page to access your account.

  **Note:** You can also create a custom login page for your users with Customer Center roles. See Creating Custom Pages for Login to Your NetSuite Account for more information.

Each type of login page displays a forgot your password link. Users can click the link, and an email is sent to the user's email address with instructions for resetting a forgotten password.

Creating Custom Pages for Login to Your NetSuite Account

NetSuite provides standard login pages for your NetSuite account. However, you can also create custom pages for login. For example, you might want to include your company's branding on the login page.

A separate login page for Customer Center roles is required. You can use the system-provided Customer Center login page for this purpose, or you can create your own custom login page, or pages.

**Note:** In 2017.2, Administrators can specify that their custom Customer Center login page be served instead of the default Customer Center login page. If you have a custom login page for your Customer Center, ensure it has been uploaded to your NetSuite File Cabinet. Then, go to Setup > Company > Company Preferences > General Preferences and scroll down to the Customer Center Login Page field. Select the filename for your Customer Center login page.

Your custom login page, and any images displayed on it, must be uploaded to the images folder in the file cabinet at Documents > Files > Images. Also, you must use the secure URL displayed on the file record in any tags you use to display content on your login page.
If you decide to create a custom login page (for Customer Center roles or for non-Customer Center roles, or for both types of roles) the login page must be **hosted in the NetSuite File Cabinet**. You can then display a link to the custom login page on a different page on your website.

**Important:** Security best practices do not allow presenting login fields to your NetSuite account in an iFrame on your web page. The following approved procedure details how to provide login access to your NetSuite account.

---

**Creating a Custom Login Page**

The following procedure describes how to create custom login pages. If you are creating a custom login page for Customer Center roles, you must know your account ID to complete this procedure. The variable in the following code example is `<ACCOUNT_ID>`.

To locate your account ID, go to Setup > Company > Setup Tasks > Company Information. The account ID field is located near the bottom of the right column.

**To create a custom login page for your NetSuite account:**

1. Create a custom login page in HTML, using the code below to display the NetSuite account login fields. Save the HTML file to your hard drive.
   - If you are creating a custom login page for non-Customer Center roles, you could, for example, name the file **NSlogin.html**. You do not have to modify the code shown below if you are creating a non-Customer Center login page.
   - If you are creating a login page for Customer Center roles, you could name the file, for example, **NSprivateLogin.html**. You must modify two lines in the sample. In each line you modify, replace the variable `<ACCOUNT_ID>` with your account ID.
     - Modify the first line (the post action link) as shown:
       ```html
       <form method="post" action="/app/login/secure/privateLogin.nl?c=<ACCOUNT_ID>"
       ```
     - Modify the href line for the Forgot your password link as shown:
       ```html
       <href="/app/login/preparepwdreset.nl?private=t&c=<ACCOUNT_ID>">
       ```

**Note:** The following code only represents the basic required fields for login to your NetSuite account. You can add content to this file, but you must use a secure URL to refer to any additional files.

```html
<!--The post action link below is for a non-Customer Center login page-->
<form method="post" action="/app/login/nllogin.nl">
<!--For a Customer Center login page, modify the post action link as specified in step 1.-->
<table border="0" cellspacing="0" cellpadding="3">
<tr>
    <td>
        Email address:<input name="email" size="30">
    </td>
</tr>
<tr>
    <td>
        Password:<input name="password" size="30" type="password">
    </td>
</tr>
</table>
```
Creating Custom Pages for Login to Your NetSuite Account

2. Go to the Images folder in the NetSuite File Cabinet (Documents > Files > Images).

3. Click Add File, and then select the appropriate HTML file for the custom login page that you created in step 1. Ensure that the "Available without login" box is selected.

4. Click Open. The HTML file for your custom login page is uploaded to the file cabinet. You can also add any additional files you want to use for content on your custom login page to this folder. Ensure that the "Available without login" box is selected for these files.

5. Determine the secure URL for your custom login page. You will use the secure URL later to display the link to your custom login page.
   a. Go to the Images folder in the NetSuite File Cabinet (Documents > Files > Images).
   b. Click Edit next to the HTML file for your custom login page.
   c. Copy the NetSuite URL that starts with https://system. You will use this URL to create a link to your login page.

6. Reference your custom login page from your website. You can now link to your custom login page from any external source by adding an href that uses the secure URL you copied in step 5.c. For example:

   `<a href="https://system.netsuite.com/....">Login Here</a>`

   Do not copy the example! Use the URL you copied in step 5.c. in your href.

   **Important:** The HTML file for the custom login page you created in step 1 must be hosted in the NetSuite File Cabinet. The external source hosting the link does not have to be in the NetSuite File Cabinet.

Security policies and contractual agreements prohibit displaying a NetSuite login page in an iFrame. For more information, see NetSuite Login Pages and iFrame Prohibition.

Customizing Login and Logout Behavior

You can customize the behavior when a user logs in to NetSuite and the behavior when a user logs out of NetSuite.
Customizing Login Page Behavior

Using a Redirect Parameter

You can redirect users, after login, to a specific landing page in the NetSuite UI. For example, you might want to have NetSuite open up on a Customer record, or a Support Case record.

To redirect a user to a particular page after login:
1. Add a redirect hidden field to the login form in your hosted HTML page, for example:
   ```html
   <input type="hidden" name="redirect" value="/app/center/card.nl?success=true" >
   ```
2. Follow the steps in the procedure in the section Creating a Custom Login Page.

Displaying an Error Message

When someone attempts to login with the wrong password or email, you can display an error on your hosted login page. This lets you maintain consistent company branding on the login page, instead of redirecting to a generic NetSuite error page.

To display an error message on your custom login page:
1. Add an error redirect hidden field to the login form in your hosted HTML page, for example:
   ```html
   <input type="hidden" name="errorredirect" value="/core/media/media.nl?id=572&c=TSTDRV1154923&h=b0c2553e7af5af87ef2&success=false" >
   ```
2. Create a separate version of your HTML login page that includes the error message, or implement conditional logic in your custom HTML login page.
3. Follow the steps in the procedure in the section Creating a Custom Login Page.

Customizing Logout Behavior

You can connect your company website's look and feel with the NetSuite application by specifying a landing page when users log out of a NetSuite center.

To specify a landing page for logout:
1. Go to Setup > Company > General Preferences
2. Click the Centers subtab to select the appropriate center.
3. Enter the URL for the Sign Out Landing Page.

NetSuite Login Pages and iFrame Prohibition

Security policies and contractual agreements prohibit displaying a NetSuite login page in an iFrame. This prohibition is documented in Secure Login Access to Your NetSuite Account.

As part of a continuing commitment to provide the most secure environment possible, since January 2015, we have been enforcing the prohibition against the use of iFrames on the following login pages:
NetSuite Login Pages and iFrame Prohibition

- `/pages/customerlogin.jsp`
  For example: https://system.netsuite.com/pages/customerlogin.jsp

- `/pages/login.jsp`
  For example: https://system.netsuite.com/pages/login.jsp

This prohibition is intended to protect against what is known as a clickjacking attack. For more information on defending against this vulnerability, see https://www.owasp.org/index.php/Clickjacking_Defense_Cheat_Sheet. This enforcement change is in accordance with best practices outlined in RFC7034 - HTTP Header Field X-Frame-Options.

To allow logins through NetSuite, you must create a login page hosted on the NetSuite secure server and display a link to this login page on a different page.

See Creating Custom Pages for Login to Your NetSuite Account for more information.
Enabling and Creating IP Address Rules

You can limit access to your company's NetSuite account by entering IP address rules. Only computers with IP addresses that match those you have entered will be permitted to access your NetSuite account. For example, you may want employees logging in to your NetSuite account from a trusted location as an additional requirement.

**Note:** To further secure the user login process, NetSuite two-factor authentication is the preferred alternative to restricting access by IP address. For more information, see Two-Factor Authentication (2FA).

**Warning:** IP addresses were designed primarily to serve host identification and addressing, thus they cannot fully serve as a reliable second factor for user authentication. Consider the following precautions, but be advised that two-factor authentication is strongly recommended.

- Only public IPv4 addresses can be used. Private IPv4 addresses cannot be used outside of your private network.
- IPv6 addresses are not supported.
- Make sure that you are the only owner of the public IPv4 address and that it is not shared among multiple ISP clients.
  
  With the increasing number of network devices, it is difficult to determine the IPv4 address of the client reliably. Increased scarcity of IPv4 addresses is leading ISPs to use Carrier-Grade NAT (CGN), Large-Scale NAT (LSN), and shorter Dynamic Host Configuration Protocol (DHCP) lease times. The client IPv4 address is not usually designated to one client, nor is it static.
- Any IP packet can be spoofed and the source-address modified or crafted.
- Any IP address being rented to you cannot be treated as a reliable authentication factor.

New users with roles that have IP address restrictions enabled are prompted to set up security questions. However, be aware that when you apply IP address restrictions, users are not prompted to answer security questions when logging in to NetSuite or when changing roles. These IP address-restricted users are only asked to answer their security questions if they forget their passwords. See the help topic Setting Up Security Questions for more information.

Inbound single sign-on access to NetSuite respects IP address restriction rules. SOAP web services and SAML Single Sign-on also respect IP Address restriction rules.

**Warning:** SuiteAnalytics Connect access to NetSuite does not respect IP address restriction rules. Users may be able to access NetSuite data through SuiteAnalytics Connect from IP addresses that they cannot use to access the NetSuite application directly.

Two-factor authentication is the preferred alternative to restricting access by IP address. For more information, see Two-Factor Authentication (2FA). However, if you still wish to restrict access to your NetSuite account by employing IP address rules, see the following sections:

- Enable the IP Address Rules Feature
- Create Company IP Address Rules
- Create Individual IP Address Rules
- Create Roles without IP Address Restrictions
- Review or Search for Access Restrictions
Enable the IP Address Rules Feature

You can restrict access at the company level or at the employee level. If you want to use IP address restrictions at the company level, check the Inherit IP Rules From Company box on employee records. Employees then will only have access to those computers you specify on the Set Up Company page. At the employee level, you can specify certain IP addresses on employee records if you want to limit an employee to a computer(s) within the company.

Note: Two-factor authentication is the preferred alternative to restricting access by IP address. For more information, see Two-Factor Authentication (2FA).

To enable the IP address rules feature:

Important: Enabling the IP Address Rules feature does not retroactively apply IP address restrictions to preexisting customized roles.

1. Go to Setup > Company > Enable Features.
2. On the Company subtab, in the Access section, check the IP Address Rules box.
3. Click Save.

Note: IP address rules may prevent users from accessing web queries of NetSuite data. For example, this issue occurs when a user with an IP address rule creates a web query and sends it to other users who are logging in from different IP addresses.

Create Company IP Address Rules

Note: Two-factor authentication is the preferred alternative to restricting access by IP address. For more information, see Two-Factor Authentication (2FA).

To create IP address rules for your company:

1. Go to Setup > Company > Company Information.
2. In the Allowed IP Addresses field, enter valid IP addresses (in dotted decimal notation) from which you want employees in your company to access your account. Each of the numbers in the four segments (the numbers between the dots) must be between 0 and 255.

Warning: Be sure that you have entered the correct IP addresses before you log out so that you and your employees can log back in.

Use the following formats:

Important: You can enter up to 4000 characters. Use shorter forms of notation to enter addresses (such as 123.45.67.80-99 or 123.45.67.80/24 in the following examples) if necessary.

- A single IP address, such as 123.45.67.89
- A range of IP addresses, with a dash and no spaces between, such as 123.45.67.80-123.45.67.99. You can use 123.45.67.80-99 to indicate the same range.
- A list of IP address separated by spaces or commas such as 123.45.67.90, 123.45.67.97,...
An IP address with full netmask, such as 123.45.67.80/255.255.255.0

**Note:** A netmask defines which bits of the IP address are valid, the example means "use the first three segments (255.255.255), but not the fourth segment (0)".

An IP address and bitmask, such as 123.45.67.80/24

**Note:** The "24" indicates the number of bits from beginning to use in the validation – the same IP addresses are valid as in the previous example (255 means 8 bits).

An IP address and mask, such as 209.209.48.32/255.255.0.0 or 209.209.48.32/16.

**Warning:** Think carefully when using this type of notation. The mask is a binary number. For example, the IP address and mask 12.34.56.78/12.34.56.78 does not indicate only one IP address is allowed. The IP address 140.34.56.78 matches the mask in this example. There are more IP addresses that match the mask than are immediately obvious.

The text "NONE" – denies access from all IP addresses.

The text "ALL" – allows all IP addresses.

3. Click **Save**.

Now, when you or other employees log in to NetSuite, if at least one rule is defined, the IP address of the computer that is being used must match the rule(s) defined. If the computer does not match the IP address rule(s) defined, login fails and a message is displayed that login is not allowed from the current IP address.

If this employee has another role with IP address restrictions, the employee can only access that role from the addresses listed on the employee record or the addresses listed at Setup > Company > Company Information when the Inherit IP Rules from Company box is checked.

### Create Individual IP Address Rules

**Note:** Two-factor authentication is the preferred alternative to restricting access by IP address. For more information, see Two-Factor Authentication (2FA).

To allow an employee access only to specific machines, you can edit the employee's record and enter one IP addresses for each computer that can be used to access NetSuite.

Employees whose records were created before the IP Address Rules feature was enabled inherit the rules you set at Setup > Company > Company Information by default.

**To create IP address rules for individual employees:**

1. Go to Lists > Employees > Employees..
2. Click **Edit** next to the employee you want set IP address rules for.
3. Click the **Access** tab.
4. Check the Inherit IP Rules from Company box to give this employee access to the IP addresses defined at Setup > Company > Company Information.

   Clear this box to allow access for this employee only at the address you enter in the IP Address Restriction field.
If you check this box and enter addresses in the IP Address Restriction field, this employee will have access to both the addresses listed at Setup > Company > Company Information and the addresses you list on this record.

5. To give this employee access to use specific machines, clear the Inherit IP Rules from Company box, and list the IP addresses in the IP Address Restriction field.

Note: Enter valid IP addresses (in dotted decimal notation) from which you want this employee to access your account. Each of the numbers in the four segments (the numbers between the dots) must be between 0 and 255.

Use the following formats:

- A single IP address, such as 123.45.67.89
- A range of IP addresses, entered with a dash and no spaces between, such as 123.45.67.80-123.45.67.99. You can use 123.45.67.80-99 to indicate the same range.
- A list of IP address separated by spaces or commas such as 123.45.67.90, 123.45.67.97,...
- An IP address with full netmask, such as 123.45.67.80/255.255.255.0

Note: A netmask defines which bits of the IP address are valid, the example means “use the first three segments (255.255.255), but not the fourth segment (0)”

- An IP address and bitmask, such as 123.45.67.80/24
  
  Note: The “24” indicates the number of bits from beginning to use in the validation – the same IP addresses are valid as in the previous example (255 means 8 bits).

- An IP address and mask, such as 209.209.48.32/255.255.0.0 (allows 209.209.*.*)
  
  Warning: Think carefully when using this type of notation. The mask is a binary number. For example, the IP address and mask 12.34.56.78/12.34.56.78 does not indicate only one IP address is allowed. The IP address 140.34.56.78 matches the mask in this example. There are more IP addresses that match the mask than are immediately obvious.

- The text "NONE“ – denies access from all IP addresses.
- The text “ALL“ – allows all IP addresses.
- If you leave the field blank, IP address restrictions are inherited from the company level.

6. Click Save.

Create Roles without IP Address Restrictions

Note: Two-factor authentication is the preferred alternative to restricting access by IP address. For more information, see Two-Factor Authentication (2FA).

You can make exceptions to your IP address rules by customizing roles. By default, all roles are restricted by the IP address rules you set at Setup > Company > Company Information and on employee records.
You can customize roles, however, to create roles that are not restricted by these rules. This way, your employees can access certain roles from anywhere and restricted roles from only the machines you specify.

**To customize a role so that it does not have IP address restrictions:**

1. Go to Setup > Users/Roles > Manage Roles.
2. Click **Customize** next to the role type you want to assign without IP rule restrictions.
3. In the **Name** field, enter or accept the name for this non-restricted role.
4. Clear the **Restrict this role by IP Address** box.
5. On the subtabs below, click the line of any permission you want to edit.
6. Change the level of permission to **View, Full, Edit, None** or **Create**.
7. Click **Done**.
8. Click **Save**.

Now, when assigning roles on the **Access** tab of employee records, you can assign this new custom role without IP address restriction. This employee can access the custom role from any computer, regardless of the IP address rules set on the employee record or at Setup > Company > Company Information.

**Review or Search for Access Restrictions**

**Note:** Two-factor authentication is the preferred alternative to restricting access by IP address. For more information, see [Two-Factor Authentication (2FA)](#).

**Review IP Address Restrictions**

To see a list of all users and review a list of the IP addresses they are restricted to using for each assigned role, go to Setup > Users/Roles > View Login Restrictions.

**Search for User Login Restrictions**

Users with the proper privileges can search for User Login Restrictions by user, role, and IP address.

**To search for user login restrictions:**

1. Go to Setup > Users/Roles > View Login Restrictions.
2. Click **Search** in the upper right corner of the page.
3. On the search page, enter your desired search parameters in the available **User, Role, and IP Addresses Allowed to Login** fields.
   - For more information on entering search parameters, see the help topic [Defining a Simple Search](#).
   - If you need help in defining filters for a simple search, see the help topic [Tips for Defining Simple Search Filters](#).
   - If you need more search criteria, check the **Use Advanced Search** box.
     If you need help, see the help topic [Defining an Advanced Search](#).
4. Click **Submit**.
Token-based Authentication (TBA)

NetSuite supports token-based authentication (TBA) a robust, industry standard-based mechanism that increases overall system security. This authentication mechanism enables client applications to use a token to access NetSuite through APIs, eliminating the need for RESTlets or web services integrations to store user credentials.

The TBA feature was built for integrations. Of all the inbound single sign-on features available for use in NetSuite, TBA is the only mechanism mature enough to use with web services and RESTlets.

In your integrations, you might need to use certain functions that require an Administrator role. Two-Factor Authentication (2FA) for Administrator roles are enforced in all accounts. You should transition integrations that require an Administrator role to use TBA rather than user credentials. See Token-based Authentication (TBA) for Integration Application Developers.

Note: For more information about using TBA with integrations, see the help topic Integration Management in the SOAP web services section of help. See also Creating Integration Records for Applications to Use TBA.

Password rotation policies in the account do not apply to tokens, making password management unnecessary for your RESTlet and web services integrations. Token-based authentication allows integrations to comply with any authentication policy that is deployed in a NetSuite account for UI login, such as SAML Single Sign-on, Inbound Single Sign-on, and Two-Factor Authentication. You can use Two-Factor Authentication (2FA) roles and roles with SAML Single Sign-on permissions with TBA.

See the following topics for more information about TBA:

- **Token-based Authentication (TBA) Tasks for Administrators**
  - Getting Started with Token-based Authentication
  - Managing TBA Tokens in the NetSuite UI
- **Token-based Authentication (TBA) for Integration Application Developers**
  - The Three-Step TBA Authorization Flow
  - The IssueToken Endpoint
- **Troubleshoot Token-based Authentication (TBA)**
- **Token-based Authentication and RESTlets**
- **Token-based Authentication and Web Services**

Token-based Authentication (TBA) Tasks for Administrators

This section provides information on tasks for administrators. See the following topics:

- **Getting Started with Token-based Authentication**
  - Enabling the Token-based Authentication Feature
  - Set Up Token-based Authentication Roles and Token-based Authentication (TBA) Permissions
  - Assigning Users to Token-based Authentication Roles
  - Creating Integration Records for Applications to Use TBA
- **Managing TBA Tokens in the NetSuite UI**
Getting Started with Token-based Authentication

To set up token-based authentication (TBA) in your NetSuite account, complete the following tasks. Each task has a separate section providing step-by-step details.

To set up TBA in your NetSuite account:

1. Enable the feature. See Enabling the Token-based Authentication Feature.
2. Set up roles. See Set Up Token-based Authentication Roles.
   See also Token-based Authentication (TBA) Permissions.
3. Assign roles to users. See Assigning Users to Token-based Authentication Roles.
4. Set up applications for token-based authentication. See Creating Integration Records for Applications to Use TBA.
5. Create user tokens. See Managing TBA Tokens in the NetSuite UI.

Note: Tokens created in your production account are not copied to your sandbox during a refresh. To test token-based authentication in your sandbox, you must create tokens in the sandbox account. Each time your sandbox is refreshed, you will need to create new tokens in the sandbox.

Enabling the Token-based Authentication Feature

Before you can begin using TBA in your account, you must enable the feature.

To enable the token-based authentication feature:

1. Go to Setup > Company > Enable Features.
2. Click the SuiteCloud subtab.
3. In the SuiteScript section, check the following boxes:
   - Client SuiteScript. Click I Agree on the SuiteCloud Terms of Service page.
   - Server SuiteScript. Click I Agree on the SuiteCloud Terms of Service page.

   Note: Enabling both the Client SuiteScript and Server SuiteScript features is required to use RESTlets with token-based authentication.

4. In the Manage Authentication section, check the Token-based Authentication box. Click I Agree on the SuiteCloud Terms of Service page.
5. Click Save.

   Note: The Manage Access Tokens link becomes available in the Settings portlet for users with Administrator role, or users with a role that has been assigned the appropriate permission. However, before users can create access tokens, you must set up roles, assign roles to users, and create integration records for applications.

After enabling the TBA feature,

- You must set up TBA roles. See Set Up Token-based Authentication Roles. See also Token-based Authentication (TBA) Permissions.
- Administrators (or users assigned the Full level of the Setup Type Integration Application permission) can create applications for use with TBA. See Creating Integration Records for Applications to Use TBA. For more detailed information, see the help topic Creating an Integration Record.
Set Up Token-based Authentication Roles

You can use TBA with those integrations that require the Administrator or other highly privileged roles. For more information, see User Access Token – Create a TBA Token and Issue Token and Revoke Token REST Services for Token-Based Authentication.

Important: For enhanced security, two-factor authentication (2FA) is mandatory for all Administrator and other highly privileged roles for access to all NetSuite accounts. This requirement applies to production, sandbox, development, and Release Preview accounts. For more information, see Authentication Overview and Mandatory Two-Factor Authentication (2FA) for NetSuite Access.

If desired, an administrator can modify existing roles to add token-based authentication permissions, then assign users to those roles as needed. If you need more information about creating or customizing roles, see:

- NetSuite Users & Roles
- NetSuite Roles Overview

Token-based Authentication (TBA) Permissions

The following token-based authentication permissions can be added to roles as appropriate.

- **Access Token Management**
  Users with this permission:
  - Can, through the NetSuite UI, create and revoke access tokens for some users with a TBA-enabled role. A user who does not have an Administrator role cannot create tokens for an Administrator.
  - **Cannot** create access tokens for their own use.
  - **Cannot** use access tokens to log in through RESTlets or web services.

- **User Access Tokens**
  Users with this permission:
  - Can, through the Manage Access Tokens link in the Settings portlet, or by calling the issuetoken endpoint, create and revoke access tokens for their own use. For more information, see User Access Token – Create a TBA Token and Issue Token and Revoke Token REST Services for Token-Based Authentication.
  - Can use access tokens to log in through RESTlets or web services.

- **Log in using Access Tokens**
  Users with this permission:
  - Can use access tokens to log in through RESTlets or web services.
  - **Cannot** create their own access tokens through a link in the Settings portlet, or by calling the issuetoken endpoint.

You must assign TBA roles to users. See Assigning Users to Token-based Authentication Roles.

Assigning Users to Token-based Authentication Roles

After modifying roles with the appropriate token-based authentication permissions, an account administrator can assign users to those roles. TBA is available for many types of NetSuite users, including customers, employees, partners, and vendors. The following is a brief procedure for assigning a role to an
existing user. If you need more information on assigning users to roles, see the help topic NetSuite Users Overview.

To assign a user to a token-based authentication role:

1. Go to the entity record for the user:
   - If the user is an employee, go to Lists > Employees > Employees.
   - If the user is not an employee, go to List > Relationships, and then click Customers, Partners, or Vendors.
2. Click Edit next to the name of the user you want to assign the token-based authentication role.
3. Click the Access tab.
4. In the Role field, select the token-based authentication role for this user.
5. Click Add.
6. Click Save.

You must set up applications for token-based authentication. See Creating Integration Records for Applications to Use TBA.

Creating Integration Records for Applications to Use TBA

Before tokens can be created and assigned to users, an Integration record must be created for each application that will use token-based authentication. Administrators or users assigned the Full level of the Setup Type Integration Application permission can create Integration records.

- For more information about the Integration record, see the help topic Integration Management.
- For more information about using token-based authentication with SOAP web services, see the help topic Token-Based Authentication Details.

To create an integration record for an application:

The following procedure briefly describes completing an Integration record. For more detailed information about the fields in this record, see the help topic Creating an Integration Record.

1. Go to Setup > Integration > Manage Integrations > New
2. Enter a Name for your application.
3. Enter a Description, if desired.
4. The application State is Enabled by default. (The other option available for selection is Blocked.) The value of this field is always specific to one NetSuite account.

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

5. Enter a Note, if desired. The value of this field is always specific to one NetSuite account.
6. On the **Authentication** tab, check (or clear) the appropriate boxes for your application:

<table>
<thead>
<tr>
<th>Field on the Authentication tab:</th>
<th>Effect when the box is checked:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Token-based Authentication (TBA)</strong></td>
<td>▪ Activates the following two TBA fields.</td>
</tr>
<tr>
<td></td>
<td>▪ Allows creation of tokens through the UI only.</td>
</tr>
<tr>
<td><strong>TBA: IssueToken Endpoint</strong></td>
<td>▪ Allows programmatic creation of tokens using the issuetoken endpoint.</td>
</tr>
<tr>
<td>For more information, see The IssueToken Endpoint.</td>
<td>▪ This box is checked for Integration records that existed before the 19.2 release.</td>
</tr>
<tr>
<td></td>
<td><strong>Important:</strong> Check this box only if it is not possible to implement the TBA: Authorization Flow in your integration.</td>
</tr>
<tr>
<td><strong>TBA: Authorization Flow</strong></td>
<td>▪ Allows creation of tokens using the three-step authorization flow.</td>
</tr>
<tr>
<td>For more information, see The Three-Step TBA Authorization Flow.</td>
<td>▪ Activates the <strong>Callback URL</strong> field.</td>
</tr>
<tr>
<td><strong>Callback URL</strong></td>
<td>▪ Enter the appropriate valid callback URL for your application.</td>
</tr>
<tr>
<td></td>
<td>▪ The callback URL is validated when you save the integration record.</td>
</tr>
<tr>
<td><strong>User Credentials</strong></td>
<td><strong>Important:</strong> New integrations should use TBA rather than user credentials.</td>
</tr>
<tr>
<td></td>
<td>▪ When creating a new integration record, this box is checked by default.</td>
</tr>
<tr>
<td></td>
<td>▪ Clear this box to ensure this application will authenticate only using tokens and not with user credentials.</td>
</tr>
</tbody>
</table>
7. Click **Save**.

The confirmation page displays the Consumer Key and Consumer Secret for this application. The application developer will need this information.

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

8. Click **List** to view all Integration records in this account.

After these basic setup tasks are complete, you are almost ready to begin using token-based authentication in your account. Users must create tokens. See **Managing TBA Tokens in the NetSuite UI**.
Managing TBA Tokens in the NetSuite UI

Managing TBA tokens in your account includes the following:

- **Creating Tokens**

  There are various methods for creating tokens. In the NetSuite UI, the method employed depends on the permission assigned to the role. For more information, see the following topics:
  - Access Token Management – Create and Assign a TBA Token
  - User Access Token – Create a TBA Token

- **Viewing, Editing, and Revoking Tokens**

  See Viewing, Editing, Creating, and Revoking TBA Tokens to open the Access Tokens list view page. Tokens can also be created by clicking New Access Token on this page.

- **Search for tokens** in your account. See Using the TBA Access Token Search Page.

Users can also create tokens without logging in to the NetSuite UI. For more information, see the following topics:

- Token-based Authentication (TBA) for Integration Application Developers
- Issue Token and Revoke Token REST Services for Token-Based Authentication

### Access Token Management – Create and Assign a TBA Token

A user who does not have an Administrator role cannot create tokens for an Administrator. Users assigned a customized role that has the Access Token Management permission can create, assign, and manage a token for other users (except tokens for an Administrator role) in the company. For example, they could assign a token to those users who are assigned a role with only the Log in using Access Tokens permission.

**Note:** Tokens created in your production account are not copied to your sandbox during a refresh. To test token-based authentication in your sandbox, you must create tokens in the sandbox account. Each time your sandbox is refreshed, you will need to create new tokens in the sandbox.

**To create and assign a TBA token:**

1. Log in as a user with the Access Token Management permission.
2. Go to Setup > Users/Roles > Access Tokens > New.

   The Access token page displays.
4. On the Access Token page:
   a. Select the **Application Name**.
   b. Select the **User**.
   c. Select the **Role**.
   d. The **Token Name** is already populated by default with a concatenation of Application Name, User, and Role. Enter your own name for this token, if desired.

5. Click **Save**.
   The confirmation page displays the Token ID and Token Secret.

   **Warning:** For security reasons, the only time the Token ID and Token Secret values are displayed is on the confirmation page. After you leave this page, these values cannot be retrieved from the system. If you lose or forget these credentials, you will need to create a new token and obtain new values.

   Treat these values as you would a password. Never share these credentials with unauthorized individuals and never send them by email.

### User Access Token – Create a TBA Token

Users assigned a role that has the **User Access Token** permission can create, assign, and manage tokens for the current user and current role.

**Note:** Tokens created in your production account are not copied to your sandbox during a refresh. To test token-based authentication in your sandbox, you must create tokens in the sandbox account. Each time your sandbox is refreshed, you will need to create new tokens in the sandbox.

**To create a token using the Manage Access Tokens link:**

1. Log in using a role with the **User Access Token** permission.
2. In the **Settings** portlet, click **Manage Access Tokens**.

   ![Manage Access Tokens](image)

   The My Access Tokens page displays, listing all the tokens for the current user in the current role.

3. Click **New My Access Token**.
The Access Token page displays.

4. On the Access Token page:
   a. Select the **Application Name**.
   b. The **Token Name** is already populated by default with a concatenation of Application Name, User, and Role. Enter your own name for this token, if desired.
5. Click **Save**.
   The confirmation page displays the Token ID and Token Secret.

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

Viewing, Editing, Creating, and Revoking TBA Tokens

You can see a list view of tokens in your system.

**To view tokens:**

   The Access Token page displays.

   Actions you can take from this page include:
   - Click **View** to open the Access Token page and review the details of a specific token.
   - Click **New Access Token** to open the Access Token page and create a new token. For more information, see Access Token Management – Create and Assign a TBA Token.
   - Click **Edit** to open the Access Token page and:
     - Edit specific details about the token, or
     - Click **Revoke** to revoke the token. For more information, see Revoking TBA Tokens in the NetSuite UI.
Open the Filters panel to select a value for **Revoked** status (All, Yes, or No).

Click **Search** at the top right corner of the Access Tokens page. For more information, see Using the TBA Access Token Search Page.

### Revoking TBA Tokens in the NetSuite UI

This section provides information about revoking a token in the NetSuite UI. For information about revoking a token programmatically, see the help topic Issue Token and Revoke Token REST Services for Token-Based Authentication.

Revoking a token makes it inactive forever, but does not remove the token from the system. The token is still accessible for auditing purposes.

**Revoke and Inactive Statuses**

- When a token is revoked, it cannot be edited, and will display with an Inactive status in list views.
- When the **Inactive** box is checked for a token, the token will display as Inactive in list views, but the token can still be edited. To make the token active again, click **Edit**, clear the **Inactive** box, and click **Save**.

### Additional Situations Under Which Tokens are Revoked

- When an application used for token-based authentication is deleted, all tokens associated with that application are revoked.
- When an administrator removes roles from an entity (an employee, a vendor, a partner, a customer, or a contact) the tokens are still active in the system. These active tokens cannot be used by the entity for log in to NetSuite (unless the administrator adds the roles back to the entity).
- When an administrator deletes an entity, (an employee, a vendor, a partner, a customer, or a contact) the associated tokens are deleted.

### Using the TBA Access Token Search Page

There are two methods of opening the Access Token Search page. One method is to click the Search link on the top right corner of a page. See the following procedure for the other method of opening the search page.

**To search for a token:**

   
   The Access Token Search page displays.
2. Enter or select from the available criteria, as appropriate.
3. click Submit.

For information on NetSuite's search capabilities, see:

- Running Searches
- Saved Searches

Token-based Authentication (TBA) for Integration Application Developers

Developers now have two options for granting tokens for applications. See the following for more information on these methods.

- The Three-Step TBA Authorization Flow
- The IssueToken Endpoint

The Three-Step TBA Authorization Flow

In 2019.2, application developers and integrators have the option to use a redirection-based authorization flow with token-based authentication. User credentials are not stored or entered into the application forms. User credentials are entered by a user into a trusted NetSuite login form as a part of the authorization flow. The redirection-based authorization flow consists of three steps:

- Step 1: Obtain an unauthorized request token on the request token URL.
- Step 2: Authorize the request token on the user authorization URL.
  
  Any authentication procedure relevant to a user (for example, a second-factor verification step) is included in this step of the authorization flow.
- Step 3: Exchange the request token for an access token on the access token URL.
With this TBA: Authorization Flow feature, integration developers begin the process to grant access tokens in their application. The request token URL generates an intermediate (unauthorized) request token. A user, for whom an access token is to be granted, authorizes the request token and explicitly consents that the application is allowed access to NetSuite data. If this step succeeds, the application exchanges the request token for an access token to be used for authentication.

This three-step authorization flow should be used for all new integrations. Developers of existing integrations currently using the issuetoken endpoint should consider migrating the integration to the three-step authorization flow.

The Administrator must create integration records for each application. See Creating Integration Records for Applications to Use TBA. The administrator must configure the callback URL on the integration record. The underlying application must have the ability to open a browser, and must be able to handle callback URLs.

**Note:** If the application does not have the ability to open a browser and handle callback URLs, developers should continue to use the issuetoken endpoint. If this is the case for your application, see The IssueToken Endpoint.

## The OAuth Authorization Header in TBA

See the following table for information on the parameters for the OAuth authorization header.

<table>
<thead>
<tr>
<th>OAuth Authorization Header Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oauth_consumer_key</td>
<td>Identifies the client. (The service attempting to access the resource.) The value of the consumer key is provided when the Integration record is created. The consumer key should be verified by the service provider.</td>
</tr>
<tr>
<td>oauth_signature_method</td>
<td>Only HMAC-SHA256 is supported.</td>
</tr>
<tr>
<td>oauth_signature</td>
<td>Constructed signature (consumer secret to be used during signing) For more information on how a signature is constructed, see The Signature. The signature should be verified by the service provider.</td>
</tr>
<tr>
<td>oauth_timestamp</td>
<td>Number of seconds passed since 1st January 1970 00:00:00 GMT Must be a positive integer Should be equal or greater than any timestamp passed in previous requests</td>
</tr>
<tr>
<td>oauth_nonce</td>
<td>Generated random string. Must be unique for all requests with the same timestamp</td>
</tr>
<tr>
<td>oauth_version</td>
<td>Optional. If present, value must be 1.0.</td>
</tr>
<tr>
<td>oauth_callback</td>
<td>An absolute URL, to which a redirect with a verification code will be performed. The callback URL should be verified. The validity of the callback URL is verified when the Integration record is saved.</td>
</tr>
<tr>
<td>OAuth Authorization Header Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>realm</td>
<td>NetSuite Account ID (company identifier).</td>
</tr>
</tbody>
</table>
| role                                 | ■ Optional.  
  ■ Indicates for which role an access token will be granted. |

⚠️ **Important:** The parameter names and values are first encoded, as per the requirements in Parameter Encoding. For more information, see https://oauth.net/core/1.0a/#encoding_parameters. The encoded parameters and values are concatenated by & (the ampersand character).

When an authorization request is successfully verified, the application should generate a Request token and a Token Secret. The following HTTP response is returned:

<table>
<thead>
<tr>
<th>Response Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oauth_token</td>
<td>An unauthorized Request Token, which will be authorized in step 2 of the flow.</td>
</tr>
<tr>
<td>oauth_secret</td>
<td>The corresponding Token Secret, to be used for signature creation in step 3 of the flow.</td>
</tr>
<tr>
<td>oauth_callback_confirmed</td>
<td>Response must be True, if the request verification was successful.</td>
</tr>
</tbody>
</table>

For more information, see [Troubleshoot Token-based Authentication (TBA)](#).

**The IssueToken Endpoint**

Available in NetSuite since 2015.1, the issuetoken endpoint is a programmatic method for creating tokens. The issuetoken authentication mechanism enables client applications to access NetSuite APIs using a token, significantly reducing the risk of compromising user credentials.

⚠️ **Important:** The three-step authorization flow should be used for all new integrations. Developers of existing integrations currently using the issuetoken endpoint should consider migrating the integration to the three-step authorization flow. See [The Three-Step TBA Authorization Flow](#) for more information.

See the following sections for more information:

- Mandatory 2FA, the IssueToken Endpoint, and nlauth_otp
- The NLAuth Authorization Header in TBA

**Mandatory 2FA, the IssueToken Endpoint, and nlauth_otp**

To accommodate the requirement for mandatory 2FA for highly privileged roles, the issuetoken endpoint was extended. The NLAuth authentication header includes an optional parameter, nlauth_otp. You can use the nlauth_otp parameter to include a one-time password (OTP) in the NLAuth header. The OTP is equivalent to the 2FA verification code provided by a user logging in to the NetSuite UI. Users can generate the necessary codes using an authenticator app, such as Google Authenticator, Microsoft Authenticator, or Okta Verify. The authentication application you choose must support OATH TOTP, the IETF RFC 6238 standard. Go to [https://tools.ietf.org/html/rfc6238](https://tools.ietf.org/html/rfc6238) to review the standard.

An authenticator app is configured for and linked to a user's email address. The verification code must be synchronized to the email address used in the NLAuth header for the integration.
Note: An authenticator app must generate the verification code included in an NLAuth header. Verification codes such as those supplied by an email, SMS message, voice call, or from a backup code are not acceptable.

The OTP is a TOTP: a time-based one-time password. Time-based means that the verification code must be generated at the time of need: when the request is sent and being authenticated. The verification code is valid for approximately a minute surrounding the time of authentication. The validity window may vary depending on the implementation.

If the NLAuth authentication header does not include an OTP for a 2FA required role, the user receives an error message that Two-Factor Authentication is required.

Supplying Verification Codes

You must provide a method to supply the verification code in the NLAuth header. There are two ways to implement a method for generating the necessary verification code. See the following sections for more information:

- Manual Method for Supplying Codes
- Automated Method for Supplying Codes

Manual Method for Supplying Codes

You can use the manual method when interaction with a human is possible. You could code a pause into your integration and ask users to supply a verification code from their authenticator app. Users must have already configured their 2FA settings in NetSuite.

Automated Method for Supplying Codes

You must use an automated method when interaction with a human is not possible. You could implement a generator of OTPs. The implementation of TOTP in NetSuite is based on RFC 6238 https://tools.ietf.org/html/rfc6238. This specification has a reference implementation.

- The code generator must store one secret key per user, that is, per email address. (When a user is configuring 2FA settings in NetSuite, the secret key is the long string of characters shown next to the QR code displayed on the Two-Factor Authentication setup page. If a user resets 2FA settings, the secret key has a different value when 2FA is configured again.

**Important:** Each implementation of an authenticator app may have a different number of digits for the verification code, and a different validity window. The validity window is the length of time that the code is valid. The NetSuite implementation accepts a six-digit verification code, and the code is valid for 30 seconds.

- If the time is not perfectly synchronized, plus or minus a few seconds should not cause a verification code to be rejected.
- OTP means one-time password. Each code can be used only a single time. OTPs cannot be reused. If two integrations hit the issuetoken endpoint with the same verification code for the same user at the same time (within 30 seconds) then the second integration will fail. To avoid this situation, the best practice is to utilize different users and roles for multiple integrations. Otherwise, you must include logic in your code that forces the client to wait 30 seconds and use the next available code.

For more information on the NLAuth authentication header and the issuetoken endpoint, see Using User Credentials for RESTlet Authentication. See also Issue Token and Revoke Token REST Services for Token-Based Authentication.
The NLAuth Authorization Header in TBA

To construct an NLAuth authorization header, use the fields described in the following table.

⚠️ **Important:** Strings must be escaped using RFC 3986. If you do not escape characters in the header, you may receive an INVALID_LOGIN_ATTEMPT error. For more information about percent encoding, see [https://tools.ietf.org/html/rfc5849#section-3.6](https://tools.ietf.org/html/rfc5849#section-3.6).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlauth_account</td>
<td>The Account ID of the NetSuite account.</td>
</tr>
<tr>
<td>nlauth_email</td>
<td>The email address with which the user logs in to NetSuite.</td>
</tr>
<tr>
<td>nlauth_signature</td>
<td>The user's password.</td>
</tr>
<tr>
<td>nlauth_role</td>
<td>The internal ID of a role with which the user is associated.</td>
</tr>
<tr>
<td>nlauth_application_id</td>
<td>The application ID of the integration associated with the RESTlet.</td>
</tr>
<tr>
<td>nlauth_otp</td>
<td>The value of the one-time password (OTP) is the same as the value of a two-factor authentication (2FA) verification code generated by an authenticator app when a user is logging in to the NetSuite UI. For more information, see Troubleshoot Token-based Authentication (TBA).</td>
</tr>
</tbody>
</table>

⚠️ **Important:** This parameter can only be used with the issuetoken endpoint.

For more information, see Troubleshoot Token-based Authentication (TBA).

Troubleshoot Token-based Authentication (TBA)

See the following sections for troubleshooting information for TBA:

- **TBA and the Login Audit Trail**
  - Track TBA Tokens and Users
  - TBA-Related Error Messages in the Login Audit Trail
- **The Signature**
  - Generate a Signature
    - Input Parameters for the Example
    - Step 1: Construct a Base String for the Signature
    - Step 2: Signature Key
    - Step 3: Signature
- **The Authorization Headers**
  - Create the Authorization Header
    - SOAP Web Services Header
    - RESTlet Header
- **The RESTlet Base String**
  - Create the RESTlet Base String Manually
  - The restletBaseString Function
TBA and the Login Audit Trail

This section covers how to track tokens and users with the Login Audit Trail and provides details about error messages you might encounter.

Track TBA Tokens and Users

You can use the Login Audit Trail to track TBA tokens and users.

To track tokens and users:

1. Go to Setup > Users/Roles > User Management > View Login Audit Trail.
2. Check the Use Advanced Search box.
3. Click the Results subtab.
4. Add the following fields: Detail, Token-based Access Token Name, and Token-based Application Name.
5. Click Submit.

The Detail column displays error messages for any token-based authentication logins with a status of Failure.

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

TBA-Related Error Messages in the Login Audit Trail

A good place to start troubleshooting TBA problems is the Detail column of the Login Audit Trail Results. RESTlets and SOAP web services have slightly different error messages, but the meaning is similar.

See the following table for information.

<table>
<thead>
<tr>
<th>RESTlet</th>
<th>SOAP Web Service</th>
<th>Problem</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>consumer_key_refused</td>
<td>consumer_key_refused</td>
<td>The application is in Blocked state on the integration record.</td>
<td>Enable the application on the integration record.</td>
</tr>
<tr>
<td>consumer_key_unknown</td>
<td>consumer_key_unknown</td>
<td>The appropriate integration record could not be found.</td>
<td>Ensure the consumer key is correct. If this error occurs with a currently enabled application, you can attempt resetting the credentials (obtain new credentials). <strong>Important:</strong> This action might break other integrations using this application. You must update the credentials in all integrations where they are used. If there is no existing integration record for this application, create one. See Creating Integration.</td>
</tr>
</tbody>
</table>

To enable the app:

1. Go to Setup > Integration > Manage Integrations > New.
2. Select the appropriate integration record, and click Edit.
3. In the State field, change Blocked to Enabled.
4. Save the record.
<table>
<thead>
<tr>
<th>RESTlet</th>
<th>SOAP Web Service</th>
<th>Problem</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>FeatureDisabled</td>
<td>FeatureDisabled</td>
<td>The Token-based Authentication feature in NetSuite is not enabled.</td>
<td>Enable the feature. See Enabling the Token-based Authentication Feature.</td>
</tr>
<tr>
<td>nonce_rejected</td>
<td></td>
<td>The nonce was not long enough.</td>
<td>Nonce must be at least six characters long. A nonce length of 20 characters is recommended.</td>
</tr>
<tr>
<td>nonce_used</td>
<td>NonceUsed</td>
<td>The combination of nonce and timestamp has already been used by this user.</td>
<td>Ensure you generate a unique nonce for every request. Do not send the same request more than one time. If you must perform the same operation, you must generate a new TBA header for each subsequent request.</td>
</tr>
<tr>
<td>parameter_rejected</td>
<td></td>
<td>The parameter was either:</td>
<td>Ensure that you: Only send OAuth parameters a single time. Send all values in the correct format. Do not send a parameter without a value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- sent twice.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- sent with a malformed value.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- sent with an empty value.</td>
<td></td>
</tr>
<tr>
<td>permission_denied</td>
<td>permission_denied</td>
<td>The entity or role is not usable.</td>
<td>This error can have many reasons.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Verify that the entity and role are both active in NetSuite.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Verify the entity has access.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Verify that the role has TBA permissions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Verify that the user has not made the role inactive on the user's View Role page.</td>
</tr>
<tr>
<td>signature_invalid</td>
<td>Invalid Signature</td>
<td>The request was not signed correctly.</td>
<td>See Generate a Signature for the correct method of signing a request.</td>
</tr>
<tr>
<td>signature_method_rejected</td>
<td>Unknown Algorithm</td>
<td>The algorithm used to create signature is not supported.</td>
<td>The most secure supported algorithm is HMAC-SHA256. For best security, you should use HMAC-SHA256. Only the HMAC-SHA256 algorithm is supported for the three-step authorization method. Currently, the HMAC-SHA1 algorithm is supported for the issuetoken endpoint, but HMAC-SHA256 is preferred.</td>
</tr>
<tr>
<td>temporary_locked</td>
<td>temporary_locked</td>
<td>The user is locked out of NetSuite.</td>
<td>The user was locked out of NetSuite after six failed login attempts. The user must wait 30 minutes to unlock access to</td>
</tr>
</tbody>
</table>
### Troubleshoot Token-based Authentication (TBA)

<table>
<thead>
<tr>
<th>RESTlet</th>
<th>SOAP Web Service</th>
<th>Problem</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>timestamp_refused</td>
<td>InvalidTimestamp</td>
<td>The timestamp of the request must be within plus or minus five (+ or – 5) minutes of the server time.</td>
<td>Ensure that:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Your computer clocks are synchronized using the NTP protocol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Requests are sent soon after generating the TBA header.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Requests are not being queued before being sent to NetSuite.</td>
</tr>
<tr>
<td>token_rejected</td>
<td>token_rejected</td>
<td>The token could not be found.</td>
<td>Ensure that the token:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Is correct.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Is active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Is a token for the correct integration application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If a token does not exist, create one. See Managing TBA Tokens in the NetSuite UI.</td>
</tr>
</tbody>
</table>

### The Signature

This section covers generating a valid signature.

**Note:** The values defined in this section are the values used in The Authorization Headers and The RESTlet Base String sections.

### Generate a Signature

Some users have difficulty constructing a valid signature. The following sections describes how to correctly create a signature and provides PHP examples for each step.

- **Input Parameters for the Example**
- **Step 1: Construct a Base String for the Signature**
- **Step 2: Signature Key**
- **Step 3: Signature**

**Note:** All encoding in TBA is percent encoding. For more information about percent encoding, go to (https://tools.ietf.org/html/rfc5849#section-3.6). The examples in this section use `php rawurlencode`.

### Input Parameters for the Example

These are the input parameters used for this example.

```php
$url = 'https://rest.netsuite.com/app/site/hosting/restlet.nl?script=6&deploy=1&customParam=someValue&testParam=someOtherValue';
// or https://webservices.netsuite.com/services/NetSuitePort_2015_2 for webservices
$httpMethod = 'POST';
$tokenKey = '2b0ce516428118bc867e9996d1b7f6de3c6234c5af6b799b73d87569f5cc';
```
$tokenSecret = 'c29a677ed7df5a458c8206565187e3d67bd73aca8e3c9d8bea1478a3eb0d295';
$consumerKey = 'ef40afdd8abac111b13825dd5e5e2ddd84f865a0dd6dcf38c28aaee6b67e4';
$consumerSecret = 'd26ad3214a4b2f23b0741c8d38392ce01c3e2e3109df6c96aeac6d8999e9ab9e8bb5';
$signatureMethod = 'HMAC-SHA256'; // or HMAC-SHA1
$nonce = 'fjaLirsICGVZw8X0pp'; // substr(str_shuffle("0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"), 0, 20);
$timestamp = '1508242306'; // time();
$version = '1.0';
$realm = '123456'; // scompid

Step 1: Construct a Base String for the Signature
The first step in creating signature is constructing a Base String. This is the only step in generating a signature which is different for SOAP web services and RESTlets.

SOAP Web Services

$baseString = rawurlencode($realm) ."&". rawurlencode($consumerKey) ."&". rawurlencode($tokenKey) ."&". rawurlencode($nonce) ."&". rawurlencode($timestamp);

Example 1. SOAP Web Services Base String Example
For SOAP web services, the creation of the Base String creation is straightforward. Use percent encoding. Parameters include: realm (accountID, also called scompid), consumer key, token key, nonce, and timestamp, with the ampersand character (&) as the delimiter.

3829855&ef40afdd8abac111b13825dd5e5e2ddd84f865a0dd6dcf38c28aaee6b67e4&zb0ce51642011d0cb8d36b69e99196d1b7f6de3c6234c5af799b73d8756f95cfe1c7f

RESTlets

$baseString = oauth_get_sbs($httpMethod, $url, array('oauth_consumer_key' => $consumerKey,
'oauth_nonce' => $nonce,
'oauth_signature_method' => $signatureMethod,
'oauth_timestamp' => $timestamp,
'oauth_token' => $tokenKey,
'oauth_version' => $version));

Example 2. RESTlets Base String Example
This RESTlets example uses the oauth library. For more information, see https://tools.ietf.org/html/rfc5849#section-3.4.1.

POST https://a2f%2Frest.net_suite.com%2Fapp%2Fs%ite%2Fhosting%2Frestlet.nl%2FcustomParam%3DsomeValue%26deploy
%3D0%26oauth_consumer_key%3DDef40afdd8abac111b13825dd5e5e2ddd84f865a0dd6dcf38c28aaee6b67e4%26oauth_nonce
%3DfjaLirsICGVZw8X0pp%26oauth_signature_method%3DHMAC-SHA256%26oauth_timestamp%3D1508242306%26oauth_token
%3D0%2Bce51642011d0cb8d36b69e99196d1b7f6de3c6234c5af799b73d8756f95cfe1c7f
%3D6%26testParam%3DsomeOtherValue

Step 2: Signature Key
The signature key is used to sign the base string in the HMAC-SHA algorithm. The key is constructed from the URL-encoded values for consumer secret and token secret, with the ampersand character (&) as the delimiter.

$key = rawurlencode($consumerSecret) .'&'. rawurlencode($tokenSecret);
Step 3: Signature

The **signature** is a base64 value of the HMAC-SHA, where the message is Base String and **key** is the key from the previous step.

```php
$signature = base64_encode(hash_hmac('sha256', $baseString, $key, true));  // or sha1
```

**Example 3. SOAP Web Services Signature**

```
76wQrUWF8i3BwFAjzNtxjF0+1xj9y2Ygsj+HVeGqY=
```

**Example 4. RESTlets Signature**

```
7mpnx1RdQn4VLsyeEwCK7jF8jGQ0b1w05MU9Kg5Rmg=
```

The Authorization Headers

This section covers creating Authorization headers. The values used in the following code samples are defined in the section **The Signature**.

Create the Authorization Header

To create the authorization header, place the correct parameter in the right place.

**Note:** For RESTlets, each parameter must be rawurlencode.

See the following sections:
- **SOAP Web Services Header**
- **RESTlet Header**

**SOAP Web Services Header**

```xml
```

**Example 5. SOAP Web Services Token Passport Example**

```
<ns:tokenPassport soap:actor="http://schemas.xmlsoap.org/soap/actor/next soap:mustUnderstand="0"
    xmlns:ns="urn:messages_2015_2.platform.webservices.netsuite.com"
    <ns:account>123456</ns:account>
    <ns:consumerKey>f40afdd8abaac111b13825dd5e5e2dddbb44f86da0dd6df38c20ae6b67e4</ns:consumerKey>
    <ns:token>2b0ce5164281b0bcdb3e69e99196b7f76de3c26234c5af799b738769055c</ns:token>
    <ns:nonce>fjaLirsICGZzw8X0p</ns:nonce>
    <ns:timestamp>1508242306</ns:timestamp>

    <ns:signature algorithm="HMAC-SHA256">76wQrUWF8i3BwFAjzNtxjF0+1xj9y2Ygsj+HVeGqY=
```

**Authentication Guide**
RESTlet Header

```php
$header = '{"Authorization": "OAuth \
    \'.realm=\'rawurlencode($realm) \
    \'.oauth_consumer_key=\'rawurlencode($consumerKey) \
    \'.oauth_token=\'rawurlencode($tokenKey) \
    \'.oauth_nonce=\'rawurlencode($nonce) \
    \'.oauth_timestamp=\'rawurlencode($timestamp) \
    \'.oauth_signature_method=\'rawurlencode($signatureMethod) \
    \'.oauth_version=\'rawurlencode($version) \
    \'.oauth_signature=\'rawurlencode($signature) \"}
```

Example 6. RESTlet Header Example

```
Authorization: OAuth realm="123456",
oauth_consumer_key="ef40afdd8babac1111b13825dd5e5e2dedd64f86d5a8dd6dcf38c20aae6b67e4",
oauth_token="2b0ce51642b1188c76b69e9919661bf66e3c62345af6b799b73d87569f5cc",
oauth_nonce="fjflrlsr2CGV2X80pg", oauth_timestamp="1508242306", oauth_signature_method="HMAC-SHA256",
oauth_version="1.0", oauth_signature="7mpNx1RdQn4VLYseEwCK7jFBjGQ0blzwDSM9Kg5Rmg%3D"
```

The RESTlet Base String

The values used in the following code samples are defined in the section The Signature.

See the following topics in this section:

- Create the RESTlet Base String Manually
- The restletBaseString Function

Create the RESTlet Base String Manually

In the following example, the Base String consists of three parts. Each step contains an image of a piece of the code to show the line numbers. To view the entire code example (without line numbers) see the following section: The restletBaseString Function.

**Note:** POST parameters are used only with content type "application/x-www-form-urlencoded". However, this content type is not allowed by RESTlets.

1. HTTP method - line 3

   **Note:** The HTTP method must be in uppercase.

```javascript
function restletBaseString($httpMethod, $url, $consumerKey, $tokenKey, $nonce, $timestamp, $version, $signatureMethod, $contentType)
```  

2. URL- lines 6-16

   - URL is taken without parameters. (lines 6-12)
Troubleshoot Token-based Authentication (TBA)

- Schema (http, https) and hostname must be in lowercase. (lines 13-15)

```php
// Include url without parameters, schema and hostname must be lowercase
if (strpos($url, '?')){
    $baseUrl = substr($url, 0, strpos($url, '?'));
    $getParams = substr($url, strpos($url, '?') + 1);
} else {
    $baseUrl = $url;
    $getParams = '';
}
$hostname = strtolower(substr($baseUrl, 0, strpos($baseUrl, '/')));
$path = substr($baseUrl, strpos($baseUrl, '/', 10));
$baseUrl = $hostname . $path;
$baseUrl .= rawurlencode($baseUrl) . '&';
```

3. Parameters - lines 19-51

- Put all OAuth, GET, and POST parameters into the array of arrays. (lines 19-37)
- Parameter names and values are urldecoded before entering into array (lines 30–34)
- The array is alphabetically sorted by parameter name. (line 40)
- The string containing all parameters is created. Each name and value is separated by the equal character (=) and each pair is separated by the ampersand character (&). Both name and value are rawurlencoded. (lines 42-50)
- The whole string containing parameters is rawurlencoded before joining with rest of the Base String (line 51)

```php
$params = array();
$params['oauth_consumer_key'] = array($consumerKey);
$params['oauth_token'] = array($tokenKey);
$params['oauth_nonce'] = array($nonce);
$params['oauth_timestamp'] = array($timestamp);
$params['oauth_signature_method'] = array($signatureMethod);
$params['oauth_version'] = array($version);

foreach (explode('&', $getParams . '&', $postParams) as $param) {
    $parsed = explode('=', $param);
    if ($parsed[0] != '') {
        $value = isset($parsed[1]) ? urldecode($parsed[1]) : '';
        if (isset($params[urldecode($parsed[0])])) {
            array_push($params[urldecode($parsed[0])], $value);
        } else {
            $params[urldecode($parsed[0])] = array($value);
        }
    }
}
// all parameters must be alphabetically sorted
ksort($params);

$paramsString = '';
foreach ($params as $key => $valueArray){
    // all values must be alphabetically sorted
    sort($valueArray);
    foreach ($valueArray as $value){
        $paramsString .= rawurlencode($key) . '=' . rawurlencode($value) . '&';
    }
}
$paramsString = substr($paramsString, 0, -1);
$baseUrl = rawurlencode($paramsString);
return $baseUrl;
```
The restletBaseString Function

```php
function restletBaseString($httpMethod, $url, $consumerKey, $tokenKey, $nonce, $timestamp, $version, $signatureMethod, $postParams){
    //http method must be upper case
    $baseString = strtoupper($httpMethod) .'&';

    //include url without parameters, schema and hostname must be lower case
    if (strpos($url, '?')){
        $baseUrl = substr($url, 0, strpos($url, '?'));
        $getParams = substr($url, strpos($url, '?') + 1);
    } else {
        $baseUrl = $url;
        $getParams = "";
    }
    $hostname = strtolower(substr($baseUrl, 0, strpos($baseUrl, '/', 10)));
    $path = substr($baseUrl, strpos($baseUrl, '/', 10));
    $baseUrl = $hostname . $path;
    $baseString .= rawurlencode($baseUrl) .'&';

    //all oauth and get params. First they are decoded, next alphabetically sorted, next each key and values is encoded and finally whole parameters are encoded
    $params = array();
    $params['oauth_consumer_key'] = array($consumerKey);
    $params['oauth_token'] = array($tokenKey);
    $params['oauth_nonce'] = array($nonce);
    $params['oauth_timestamp'] = array($timestamp);
    $params['oauth_signature_method'] = array($signatureMethod);
    $params['oauth_version'] = array($version);

    foreach (explode('&', $getParams .'&'. $postParams) as $param) {
        $parsed = explode('=', $param);
        if ($parsed[0] != "") {
            $value = isset($parsed[1]) ? urldecode($parsed[1]) : "";
            if (isset($params[urldecode($parsed[0])])) {
                array_push($params[urldecode($parsed[0])], $value);
            } else {
                $params[urldecode($parsed[0])] = array($value);
            }
        }
    }

    //all parameters must be alphabetically sorted
    ksort($params);

    $paramString = "";
    foreach ($params as $key => $valueArray){
        //all values must be alphabetically sorted
        sort($valueArray);
        foreach ($valueArray as $value){
            $paramString .= rawurlencode($key) . '='. rawurlencode($value) .'&';
        }
    }
    $paramString = substr($paramString, 0, -1);
    $baseString .= rawurlencode($paramString);
}
```
return $baseString;
}

See also:

- Troubleshoot Token-based Authentication (TBA)
- TBA and the Login Audit Trail
- The Signature
- The Authorization Headers

Token-based Authentication and RESTlets

The following details about using token-based authentication with RESTlets (TBA with RESTlets) are provided here for your convenience. For more information, see Authentication for RESTlets.

**Note:** Web Services Only roles are only for access to NetSuite through web services. Roles with the Web Services Only restriction will not work with RESTlets.

For more information and examples, see the following topics:

- Authentication for RESTlets, especially:
  - Setting up Token-based Authentication for a RESTlet integration
  - Using User Credentials for RESTlet Authentication
- Issue Token and Revoke Token REST Services for Token-Based Authentication

Calling a RESTlet

Follow the OAuth 1.0 specification to generate a token. For more information and an example, see Required Data for Using TBA with RESTlets.

Token-based Authentication and Web Services

Token-based Authentication (TBA) supports web services.

Token-based authentication removes the problems associated with password expiration from SOAP web services authentication. Client applications can access web services using a token, significantly reducing the risk of compromising user credentials. For more information, see the help topic Integration Management.

For guidance on adapting an integration to include TBA credentials and to see an example that includes code snippets and SOAP headers, see the help topic Token-Based Authentication Details.

With TBA, you use the TokenPassport complex type to send credentials. The TokenPassport references the TokenPassportSignature complex type, another important element in the token-based authentication process. See the help topic TokenPassport Complex Type.

For more information about using token-based authentication with web services, see the following topics in the SOAP web services section of help:

- Requirements for Using Token-Based Authentication
- Regenerating a Consumer Key and Secret
- SOAP Web Services Governance for Token-Based Authentication
- Issue Token and Revoke Token REST Services for Token-Based Authentication
Two-Factor Authentication (2FA)

Two-factor authentication (2FA) allows enforcement of a second level of security for logging in to the NetSuite user interface. Using 2FA can protect your company from unauthorized access to data.

Two-factor authentication requires that users log in to the NetSuite UI with:

- NetSuite user credentials: their email address and password.
- A verification code supplied by one of the following:
  - An authentication application that complies with OATH TOTP. The app generates a time-based verification code for each login.
  - A phone that can receive verification codes by Short Message Service (SMS) message or by voice call.
  - A verification code from a list of backup codes.

Each verification code is a unique series of numbers valid for a limited time, and only for a single login.

Users can specify how they wish to receive verification codes when they set up their 2FA preferences. To read 2FA help topics available to users, see the help topic Logging In Using Two-Factor Authentication (2FA).

See the following for more information:
- What Administrators Need to Know About 2FA
- Benefits of 2FA in Your NetSuite Account

What Administrators Need to Know About 2FA

- As of 2018.1, certain roles with highly privileged permissions require 2FA. See the help topic Permissions Requiring Two-Factor Authentication (2FA).
- New users are prompted to set up security questions when they first log in to NetSuite. However, be aware that users logging in with a 2FA authentication required role are not prompted to answer security questions. The level of security provided by 2FA authentication is greater than that provided by security questions. Users logging in with 2FA roles are only asked to answer their security questions if they forget their passwords. See the help topic Setting Up Security Questions for more information.
- 2FA is not compatible with web services or SuiteAnalytics Connect. To use web services or SuiteAnalytics Connect, you must be logged in with a role that does not require 2FA. If you want to use RESTlets or web services with a highly privileged role, use Token-based Authentication. See Token-based Authentication (TBA) for more information.
- If a role is designated as a SAML Single Sign-on (SSO) role, the SAML authentication requirement takes precedence, and the 2FA requirement is ignored.

Note: The NetSuite feature that required RSA SecurID tokens is no longer available for purchase. Customers requiring 2FA for account access should use the 2FA solution built in to NetSuite.

Benefits of 2FA in Your NetSuite Account

The benefits of 2FA include:
No special licensing is required. (No cost.)
No special tokens are required. (No cost.)
Access is supported for the NetSuite UI and NetSuite Mobile applications.
Little maintenance is required of administrators. After being assigned to a 2FA authentication required role, users configure their own 2FA settings and manage their own devices in NetSuite.
Self-service user setup: pages in the NetSuite UI guide users through setting up primary and secondary 2FA authentication methods, and provides users with backup codes.
2FA works with all non-customer center roles, including contacts.
The user’s 2FA setup is shared across all NetSuite accounts and for all companies to which they have access.
There are several authentication options available for users, and users can switch between these options when they log in:
- The Authenticator App option should be the user’s primary authentication method because it is always available. Even when the phone is offline, the app is not. When a user cannot receive an SMS message or a voice call, the authenticator app can generate a verification code. For a list of third-party authentication applications, see the help topic Supported Authenticator Apps. See also Troubleshoot Authenticator Apps.
- The SMS and Voice Call options let users specify their preferred delivery method for verification codes: SMS message or voice call. Users only need to set up a phone number in NetSuite and specify how they prefer to receive verification codes. If necessary, administrators can verify which delivery methods are available in their country. See Supported Countries: SMS and Voice Call.

Note: For information on other authentication methods available in NetSuite, see Authentication Overview

Managing Two-Factor Authentication

Administrators do not have to enable a feature to use 2FA in a NetSuite account. You do not have to purchase or upload tokens. Setup required of administrators is minimal. You can begin using 2FA in your NetSuite account whenever you want to get started. Account administrators, or other users with the Two-Factor Authentication base permission, must designate roles as 2FA authentication required. Users who are assigned to 2FA-required roles must set up their authenticator applications and phone numbers in NetSuite.
**Important:** 2FA is mandatory for the Administrator role and other roles with highly privileged permissions. These roles are indicated in Mandatory 2FA columns on the Two-Factor Authentication Roles page. For a list of roles that are considered highly privileged, see the help topic Permissions Requiring Two-Factor Authentication (2FA).

---

### Two-Factor Authentication Roles

<table>
<thead>
<tr>
<th>Role</th>
<th>MANDATORY 2FA for UI</th>
<th>MANDATORY 2FA for Integrations</th>
<th>TWO-FACTOR AUTHENTICATION REQUIRED</th>
<th>Duration of Trusted Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIR Clerk</td>
<td>Not required</td>
<td></td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
<tr>
<td>AIR Clerk</td>
<td>Not required</td>
<td></td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
<tr>
<td>Accountant</td>
<td>Not required</td>
<td></td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
<tr>
<td>Accountant (Reviewer)</td>
<td>Not required</td>
<td></td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
<tr>
<td>Administrator</td>
<td>✓</td>
<td>✓</td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
<tr>
<td>Advanced Partner Center</td>
<td>Not required</td>
<td></td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
<tr>
<td>Bookkeeper</td>
<td>Not required</td>
<td></td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
<tr>
<td>Buyer</td>
<td>Not required</td>
<td></td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
<tr>
<td>CEO</td>
<td>Not required</td>
<td></td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
<tr>
<td>CFO</td>
<td>Not required</td>
<td></td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
<tr>
<td>Chief People Officer (CPO)</td>
<td>Not required</td>
<td></td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
<tr>
<td>Consultant</td>
<td>Not required</td>
<td></td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
<tr>
<td>Employee Center</td>
<td>Not required</td>
<td></td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
<tr>
<td>Engineer</td>
<td>Not required</td>
<td></td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
<tr>
<td>Engineering Manager</td>
<td>Not required</td>
<td></td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
<tr>
<td>Full Access</td>
<td>✓</td>
<td>✓</td>
<td>2FA authentication required</td>
<td>Per session</td>
</tr>
</tbody>
</table>

---

**Required 2FA Tasks**

The following are required tasks for managing two-factor authentication (2FA) in a NetSuite account. These tasks can be completed by account administrators and by other users that have the Two-Factor Authentication base permission.

- For roles that you want to restrict as 2FA roles, designate the role as 2FA authentication required. See Designate Two-Factor Authentication Roles.
- When using 2FA, after administrators designate roles and assign them to users, the users:
  - Are sent a verification code by email during the initial login attempt to a 2FA role.
  - Must set up 2FA preferences, select a primary authentication method, and should select a secondary authentication method. See the following for help written for users: Set up Your Preferences for Two-Factor Authentication (2FA).
    - To receive verification codes using an Authenticator App, users must set up an authenticator application. For a list of supported authenticator apps, see the help topic Supported Authenticator Apps.
    - To receive verification codes by phone, users must register a phone number in NetSuite, which is tied to the user’s email address.
    - Users are provided ten backup codes, to be used when they are not able to receive a verification code through their authenticator app, SMS message, or a voice call.
Each time a user logs in to NetSuite, they must enter an email address and password. If the role is a 2FA authentication required role, the user must enter a verification code obtained from an authenticator app, or from an SMS message or voice call. Each verification code is a unique series of numbers valid for a limited time, and only for a single login. During setup, users are also supplied with backup codes that can also be used for 2FA access.

**Tip:** Are your users planning a trip to a location where they do not have phone service? Authenticator apps can provide a verification code even when there is no phone service. They should also take their backup codes with them. Remind them to keep their backup codes secure. Do not store backup codes with the login device.

For help written for users, see the help topic Logging In Using Two-Factor Authentication (2FA).

### Designate Two-Factor Authentication Roles

**Note:** The NetSuite feature that required RSA SecurID tokens is no longer available for purchase. Customers requiring 2FA for account access should use the 2FA solution built in to NetSuite.

An account administrator or another user with the **Two-Factor Authentication base** permission can use the Two-Factor Authentication Roles page to indicate roles that require 2FA for login. Each 2FA role can be configured to specify how often users with that role should be presented with the 2FA challenge. The default is per session, and the Duration of Trusted Device column includes values for hours (4, 6, 8, 12) and days (1–30). The value specified in the Duration of Trusted Device column works in conjunction with the devices users indicate as trusted devices. See Users and Trusted Devices for Two-Factor Authentication for more information.

**Important:** The 2FA authentication required designation can be applied to most roles, including Employee Center, Partner Center, and Vendor Center roles, but not to Customer Center roles.

To designate two-factor authentication roles:

2FA is mandatory for the Administrator role and other roles with highly privileged permissions. These roles are indicated in the Mandatory 2FA columns on the Two-Factor Authentication Roles page. For more information, see the help topic Permissions Requiring Two-Factor Authentication (2FA).

1. Go to Setup > Users/Roles > Two-Factor Authentication Roles.
2. Select **2FA authentication required** from the list in the Two-Factor Authentication Required column for any role that you want 2FA to be required.
3. In the **Duration of Trusted Device** column, accept the default (Per session) or select the length of time before a device a user has marked as trusted will be subject to a two-factor authentication request.

4. **Click Submit**.

**Note:** The Two-Factor Authentication feature is not compatible with web services or SuiteAnalytics Connect. To use web services or SuiteAnalytics Connect, you must be logged in with a role that does not require 2FA. If you want to use RESTlets or web services with a highly privileged role, use Token-based Authentication. See **Token-based Authentication (TBA)** for more information.

If you need more information about setting up access or roles in NetSuite, see the help topics [NetSuite Roles Overview](#) and [NetSuite Access Overview](#).

### Users and Trusted Devices for Two-Factor Authentication

**Note:** The NetSuite feature that required RSA SecurID tokens is no longer available for purchase. Customers requiring 2FA for account access should use the 2FA solution built in to NetSuite.

Users with 2FA authentication required roles can specify devices as trusted when logging in to the 2FA role. Marking a device as trusted works in conjunction with the value specified by the Administrator in the **Duration of Trusted Device** column for a particular role.

For example, a role has been designated as 2FA required, and the value for Duration of Trusted Device has been set to 30 days. The next time a user with this role logs in, they can choose whether to check the **Trust this device** box.
In cases where a user has access to more than one company, and the user’s role is 2FA authentication required, marking a device as trusted makes that device trusted across all companies to which the user has access.

The user is in complete control of whether devices are considered trusted. In this example, the user could check the Trust this device box and then would not be presented with a 2FA challenge for 30 days when logging in to NetSuite from this device.

After a user has marked a device as trusted, the user can modify that choice on the Manage Trusted Devices page.

For example, this user marked a device as trusted. That is, the user previously chose not to be asked for a Two-Factor Authentication (2FA) verification code on this device.

The user can reverse that choice by selecting a Restore 2FA required... option. If the 2FA required is restored for a device, the user must use a 2FA authenticator app, phone, or a backup code to log in.

There is a link on the Settings portlet to access the Manage Trusted Devices page.

2FA in the NetSuite Application

View a related video.
Two-Factor Authentication, or 2FA, is available for all companies using NetSuite in all their NetSuite accounts as a method of improving security. 2FA can help you to comply with IT security standards and regulations, using phones your users already have. 2FA is not tied to a single company in NetSuite. As long as the user's session remains valid, the user will not be asked again for a verification code when they switch between roles, even when switching between roles in different companies.

The option to use an Authenticator App for 2FA is available in your account, and is the recommended option for users with roles that are designated as 2FA authentication required. It is not always possible for users to receive an SMS message or voice call. Authenticator apps are always available for generating verification codes. See the help topic Supported Authenticator Apps for more information on choosing an app.

To use 2FA, account administrators (or other users with the permission Two-Factor Authentication base) must designate specific roles as 2FA authentication required roles. See Designate Two-Factor Authentication Roles for more information.

Each user assigned to a 2FA role designated as 2FA authentication required must set up an authenticator application or a phone number in NetSuite. The user's phone number is linked to the email address they use to log in to the NetSuite UI.

After a role has been designated as 2FA authentication required, a user assigned to that role receives an email the first time they attempt to login to the 2FA role. The email contains instructions and a verification code for initial login.

After completing the initial login to a 2FA role, a wizard opens allowing the user to select their preferred options for generating 2FA verification codes.

See the help topic Logging In Using Two-Factor Authentication (2FA) for documentation written for those users who are not administrators.

### Reset a User’s 2FA Settings

Account administrators can reset 2FA settings for users who may be unable to log in and reset them on their own.

**Note:** 2FA is not restricted to a single company. An account administrator can only reset a user’s 2FA settings when the user has access to the same company accounts that the administrator manages. If the administrator can reset a user’s password, the administrator should also have the ability to reset the user’s 2FA settings.

**To reset a user’s 2FA settings:**

1. As an Administrator, go to Setup > Users/Roles > Two-Factor Authentication > Two-Factor Reset Tool.
2. On the Reset 2FA Settings page, enter the **Email Address** of the user whose 2FA settings you want to reset.
3. Click **Reset**.

**Important:** If you receive an error message that the 2FA settings cannot be reset, it usually indicates that you do not have management over all the accounts that the user has access to under that email address. Click **Go Back** and click **Cancel**. Contact NetSuite Customer Support for assistance with resetting the 2FA settings for the user.

A confirmation page states that the registered 2FA devices have been successfully reset.
Supported Countries: SMS and Voice Call

Although the phone number setup required for users is fairly straightforward, you or your users might have questions about the supported delivery methods available in your country for receiving verification codes.

The following table lists supported countries and the supported delivery methods for access to NetSuite.

<table>
<thead>
<tr>
<th>Country</th>
<th>Supported Delivery Methods</th>
<th>Country Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afghanistan</td>
<td>SMS</td>
<td>+93</td>
</tr>
<tr>
<td>Åland Islands (Finland)</td>
<td>SMS, Voice call</td>
<td>+358</td>
</tr>
<tr>
<td><strong>Note:</strong> When setting up your phone number, Åland Islands is at the end of the list of countries in step 1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Albania</td>
<td>SMS, Voice call</td>
<td>+355</td>
</tr>
<tr>
<td>Algeria</td>
<td>SMS</td>
<td>+213</td>
</tr>
<tr>
<td>American Samoa</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>Area Code: 684</td>
<td></td>
</tr>
<tr>
<td>Andorra</td>
<td>SMS, Voice call</td>
<td>+376</td>
</tr>
<tr>
<td>Angola</td>
<td>SMS</td>
<td>+244</td>
</tr>
<tr>
<td>Anguilla</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>Area Code: 264</td>
<td></td>
</tr>
<tr>
<td>Antigua and Barbuda</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>Area Code: 268</td>
<td></td>
</tr>
<tr>
<td>Argentina</td>
<td>SMS, Voice call</td>
<td>+54</td>
</tr>
<tr>
<td>Armenia</td>
<td>SMS, Voice call</td>
<td>+374</td>
</tr>
<tr>
<td>Aruba</td>
<td>SMS</td>
<td>+297</td>
</tr>
<tr>
<td>Ascension</td>
<td>SMS</td>
<td>+247</td>
</tr>
<tr>
<td>Australia</td>
<td>SMS, Voice call</td>
<td>+61</td>
</tr>
<tr>
<td>Austria</td>
<td>SMS, Voice call</td>
<td>+43</td>
</tr>
<tr>
<td>Azerbaijan</td>
<td>SMS, Voice call</td>
<td>+994</td>
</tr>
<tr>
<td>Country</td>
<td>Supported Delivery Methods</td>
<td>Country Code</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Bahamas</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area Code: 242</td>
</tr>
<tr>
<td>Bahrain</td>
<td>SMS</td>
<td>+973</td>
</tr>
<tr>
<td>Bangladesh</td>
<td>SMS</td>
<td>+880</td>
</tr>
<tr>
<td>Barbados</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area Code: 246</td>
</tr>
<tr>
<td>Belarus</td>
<td>SMS</td>
<td>+375</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Belgium</td>
<td>SMS</td>
<td>+32</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Belize</td>
<td>SMS</td>
<td>+501</td>
</tr>
<tr>
<td>Benin</td>
<td>SMS</td>
<td>+229</td>
</tr>
<tr>
<td>Bermuda</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area Code: 441</td>
</tr>
<tr>
<td>Bhutan</td>
<td>SMS</td>
<td>+975</td>
</tr>
<tr>
<td>Bolivia</td>
<td>SMS</td>
<td>+591</td>
</tr>
<tr>
<td>Bosnia and Herzegovina</td>
<td>SMS</td>
<td>+387</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Botswana</td>
<td>SMS</td>
<td>+267</td>
</tr>
<tr>
<td>Brazil</td>
<td>SMS</td>
<td>+55</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>British Virgin Islands</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area Code: 284</td>
</tr>
<tr>
<td>Brunei</td>
<td>SMS</td>
<td>+673</td>
</tr>
<tr>
<td>Bulgaria</td>
<td>SMS</td>
<td>+359</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Burkina Faso</td>
<td>SMS</td>
<td>+226</td>
</tr>
<tr>
<td>Burundi</td>
<td>SMS</td>
<td>+257</td>
</tr>
<tr>
<td>Cambodia</td>
<td>SMS</td>
<td>+855</td>
</tr>
<tr>
<td>Cameroon</td>
<td>SMS</td>
<td>+237</td>
</tr>
<tr>
<td>Canada</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Multiple Area Codes</td>
</tr>
<tr>
<td>Canary Islands</td>
<td>SMS</td>
<td>+3491</td>
</tr>
<tr>
<td>Country</td>
<td>Supported Delivery Methods</td>
<td>Country Code</td>
</tr>
<tr>
<td>---------------------------------------------</td>
<td>----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Cape Verde</td>
<td>SMS</td>
<td>+238</td>
</tr>
<tr>
<td>Caribbean Netherlands (Netherlands Antilles)</td>
<td>SMS</td>
<td>+599</td>
</tr>
<tr>
<td>Cayman Islands</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>Area Code: 345</td>
<td></td>
</tr>
<tr>
<td>Central African Republic</td>
<td>SMS</td>
<td>+236</td>
</tr>
<tr>
<td>Chad</td>
<td>SMS</td>
<td>+235</td>
</tr>
<tr>
<td>Chile</td>
<td>SMS</td>
<td>+56</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>China</td>
<td>SMS</td>
<td>+86</td>
</tr>
<tr>
<td>Christmas Island</td>
<td>SMS</td>
<td>+61</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Cocos (Keeling) Islands</td>
<td>SMS</td>
<td>+61</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Colombia</td>
<td>SMS</td>
<td>+57</td>
</tr>
<tr>
<td>Comoros</td>
<td>SMS</td>
<td>+269</td>
</tr>
<tr>
<td>Congo, Democratic People's Republic</td>
<td>SMS</td>
<td>+243</td>
</tr>
<tr>
<td>Congo, Republic of</td>
<td>SMS</td>
<td>+242</td>
</tr>
<tr>
<td>Costa Rica</td>
<td>SMS</td>
<td>+506</td>
</tr>
<tr>
<td>Côte d'Ivoire (Ivory Coast)</td>
<td>SMS</td>
<td>+225</td>
</tr>
<tr>
<td>Croatia</td>
<td>SMS</td>
<td>+385</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Cuba</td>
<td>SMS</td>
<td>+53</td>
</tr>
<tr>
<td>Cyprus</td>
<td>SMS</td>
<td>+357</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Czech Republic</td>
<td>SMS</td>
<td>+420</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Denmark</td>
<td>SMS</td>
<td>+45</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Djibouti</td>
<td>SMS</td>
<td>+253</td>
</tr>
</tbody>
</table>

**Note:** When setting up your phone number, in step 1, select Spain (+34) as the country. Type 91 then type your phone number.
<table>
<thead>
<tr>
<th>Country</th>
<th>Supported Delivery Methods</th>
<th>Country Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dominica</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area Code: 767</td>
</tr>
<tr>
<td>Dominican Republic</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area Code: 809</td>
</tr>
<tr>
<td>East Timor</td>
<td>SMS</td>
<td>+670</td>
</tr>
<tr>
<td>Note: When setting up your phone number, this country is Timor-Leste in the list in step 1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ecuador</td>
<td>SMS</td>
<td>+593</td>
</tr>
<tr>
<td>Egypt</td>
<td>SMS</td>
<td>+20</td>
</tr>
<tr>
<td>El Salvador</td>
<td>SMS</td>
<td>+503</td>
</tr>
<tr>
<td>Equatorial Guinea</td>
<td>SMS</td>
<td>+240</td>
</tr>
<tr>
<td>Eritrea</td>
<td>SMS</td>
<td>+291</td>
</tr>
<tr>
<td>Estonia</td>
<td>SMS</td>
<td>+372</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Ethiopia</td>
<td>SMS</td>
<td>+251</td>
</tr>
<tr>
<td>Falkland Islands</td>
<td>SMS</td>
<td>+500</td>
</tr>
<tr>
<td>Faroe Islands</td>
<td>SMS</td>
<td>+298</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Fiji</td>
<td>SMS</td>
<td>+679</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Finland</td>
<td>SMS</td>
<td>+358</td>
</tr>
<tr>
<td>(including the Åland Islands)</td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>France</td>
<td>SMS</td>
<td>+33</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>French Guiana</td>
<td>SMS</td>
<td>+594</td>
</tr>
<tr>
<td>French Polynesia</td>
<td>SMS</td>
<td>+689</td>
</tr>
<tr>
<td>Gabon</td>
<td>SMS</td>
<td>+241</td>
</tr>
<tr>
<td>Gambia</td>
<td>SMS</td>
<td>+220</td>
</tr>
<tr>
<td>Georgia</td>
<td>SMS</td>
<td>+995</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Germany</td>
<td>SMS</td>
<td>+49</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Country</td>
<td>Supported Delivery Methods</td>
<td>Country Code</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Ghana</td>
<td>SMS</td>
<td>+233</td>
</tr>
<tr>
<td>Gibraltar</td>
<td>SMS</td>
<td>+350</td>
</tr>
<tr>
<td>Greece</td>
<td>SMS</td>
<td>+30</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Greenland</td>
<td>SMS</td>
<td>+299</td>
</tr>
<tr>
<td>Grenada</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>Area Code: 437</td>
<td></td>
</tr>
<tr>
<td>Guadeloupe</td>
<td>SMS</td>
<td>+590</td>
</tr>
<tr>
<td>Guam</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>Area Code: 671</td>
<td></td>
</tr>
<tr>
<td>Guatemala</td>
<td>SMS</td>
<td>+502</td>
</tr>
<tr>
<td>Guernsey (UK)</td>
<td>SMS</td>
<td>+44</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Guinea</td>
<td>SMS</td>
<td>+224</td>
</tr>
<tr>
<td>Guinea-Bissau</td>
<td>SMS</td>
<td>+245</td>
</tr>
<tr>
<td>Guyana</td>
<td>SMS</td>
<td>+592</td>
</tr>
<tr>
<td>Haiti</td>
<td>SMS</td>
<td>+509</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Honduras</td>
<td>SMS</td>
<td>+504</td>
</tr>
<tr>
<td>Hong Kong</td>
<td>SMS</td>
<td>+852</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Hungary</td>
<td>SMS</td>
<td>+36</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Iceland</td>
<td>SMS</td>
<td>+354</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>India</td>
<td>SMS</td>
<td>+91</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Indonesia</td>
<td>SMS</td>
<td>+62</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Iran</td>
<td>SMS</td>
<td>+98</td>
</tr>
<tr>
<td>Iraq</td>
<td>SMS</td>
<td>+964</td>
</tr>
<tr>
<td>Ireland</td>
<td>SMS</td>
<td>+353</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Country</td>
<td>Supported Delivery Methods</td>
<td>Country Code</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Isle of Man</td>
<td>SMS, Voice call</td>
<td>+44</td>
</tr>
<tr>
<td>(United Kingdom)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Israel</td>
<td>SMS, Voice call</td>
<td>+972</td>
</tr>
<tr>
<td>Italy</td>
<td>SMS, Voice call</td>
<td>+39</td>
</tr>
<tr>
<td>Jamaica</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area Code: 876</td>
</tr>
<tr>
<td>Japan</td>
<td>SMS, Voice call</td>
<td>+81</td>
</tr>
<tr>
<td>Jersey</td>
<td>SMS, Voice call</td>
<td>+44</td>
</tr>
<tr>
<td>(United Kingdom)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jordan</td>
<td>SMS</td>
<td>+962</td>
</tr>
<tr>
<td>Kazakhstan</td>
<td>SMS, Voice call</td>
<td>+7</td>
</tr>
<tr>
<td>Kenya</td>
<td>SMS, Voice call</td>
<td>+254</td>
</tr>
<tr>
<td>Kosovo</td>
<td>SMS</td>
<td>+883</td>
</tr>
<tr>
<td>Kuwait</td>
<td>SMS, Voice call</td>
<td>+965</td>
</tr>
<tr>
<td>Kyrgyzstan</td>
<td>SMS, Voice call</td>
<td>+996</td>
</tr>
<tr>
<td>Laos</td>
<td>SMS</td>
<td>+856</td>
</tr>
<tr>
<td>Latvia</td>
<td>SMS, Voice call</td>
<td>+371</td>
</tr>
<tr>
<td>Lebanon</td>
<td>SMS</td>
<td>+961</td>
</tr>
<tr>
<td>Lesotho</td>
<td>SMS</td>
<td>+266</td>
</tr>
<tr>
<td>Liberia</td>
<td>SMS</td>
<td>+231</td>
</tr>
<tr>
<td>Libya</td>
<td>SMS</td>
<td>+218</td>
</tr>
<tr>
<td>Liechtenstein</td>
<td>SMS, Voice call</td>
<td>+423</td>
</tr>
<tr>
<td>Lithuania</td>
<td>SMS, Voice call</td>
<td>+370</td>
</tr>
<tr>
<td>Luxembourg</td>
<td>SMS</td>
<td>+352</td>
</tr>
<tr>
<td>Country</td>
<td>Supported Delivery Methods</td>
<td>Country Code</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Macau</td>
<td>Voice call</td>
<td>+853</td>
</tr>
<tr>
<td>Macedonia</td>
<td>SMS</td>
<td>+389</td>
</tr>
<tr>
<td>Madagascar</td>
<td>SMS</td>
<td>+261</td>
</tr>
<tr>
<td>Malawi</td>
<td>SMS</td>
<td>+265</td>
</tr>
<tr>
<td>Malaysia</td>
<td>SMS</td>
<td>+60</td>
</tr>
<tr>
<td>Maldives</td>
<td>SMS</td>
<td>+960</td>
</tr>
<tr>
<td>Mali</td>
<td>SMS</td>
<td>+223</td>
</tr>
<tr>
<td>Malta</td>
<td>SMS</td>
<td>+356</td>
</tr>
<tr>
<td>Marshall Islands</td>
<td>SMS</td>
<td>+692</td>
</tr>
<tr>
<td>Martinique</td>
<td>SMS</td>
<td>+596</td>
</tr>
<tr>
<td>Mauritania</td>
<td>SMS</td>
<td>+222</td>
</tr>
<tr>
<td>Mauritius</td>
<td>SMS</td>
<td>+230</td>
</tr>
<tr>
<td>Mayotte</td>
<td>SMS</td>
<td>+262</td>
</tr>
<tr>
<td>Mexico</td>
<td>SMS</td>
<td>+52</td>
</tr>
<tr>
<td>Micronesia</td>
<td>SMS</td>
<td>+691</td>
</tr>
<tr>
<td>Moldova</td>
<td>SMS</td>
<td>+373</td>
</tr>
<tr>
<td>Monaco</td>
<td>SMS</td>
<td>+377</td>
</tr>
<tr>
<td>Mongolia</td>
<td>SMS</td>
<td>+976</td>
</tr>
<tr>
<td>Montenegro</td>
<td>SMS</td>
<td>+382</td>
</tr>
<tr>
<td>Montserrat</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>Area Code: 664</td>
<td></td>
</tr>
<tr>
<td>Morocco</td>
<td>SMS</td>
<td>+212</td>
</tr>
<tr>
<td>Mozambique</td>
<td>SMS</td>
<td>+258</td>
</tr>
<tr>
<td>Myanmar</td>
<td>SMS</td>
<td>+95</td>
</tr>
<tr>
<td>Country</td>
<td>Supported Delivery Methods</td>
<td>Country Code</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Namibia</td>
<td>SMS</td>
<td>+264</td>
</tr>
<tr>
<td>Nepal</td>
<td>SMS</td>
<td>+977</td>
</tr>
<tr>
<td>Netherlands</td>
<td>SMS</td>
<td>+31</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Netherlands Antilles (Caribbean Netherlands)</td>
<td>SMS</td>
<td>+599</td>
</tr>
<tr>
<td>New Caledonia</td>
<td>SMS</td>
<td>+687</td>
</tr>
<tr>
<td>New Zealand</td>
<td>SMS</td>
<td>+64</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Nicaragua</td>
<td>SMS</td>
<td>+505</td>
</tr>
<tr>
<td>Niger</td>
<td>SMS</td>
<td>+227</td>
</tr>
<tr>
<td>Nigeria</td>
<td>SMS</td>
<td>+234</td>
</tr>
<tr>
<td>North Korea</td>
<td>SMS</td>
<td>+850</td>
</tr>
<tr>
<td>Northern Mariana Islands</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>Area Code: 670</td>
<td></td>
</tr>
<tr>
<td>Norway</td>
<td>SMS</td>
<td>+47</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Oman</td>
<td>SMS</td>
<td>+968</td>
</tr>
<tr>
<td>Pakistan</td>
<td>SMS</td>
<td>+92</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Palau</td>
<td>SMS</td>
<td>+680</td>
</tr>
<tr>
<td>Palestinian Territory (Palestine)</td>
<td>SMS</td>
<td>+970</td>
</tr>
<tr>
<td>Panama</td>
<td>SMS</td>
<td>+507</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Papua New Guinea</td>
<td>SMS</td>
<td>+675</td>
</tr>
<tr>
<td>Paraguay</td>
<td>SMS</td>
<td>+595</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Peru</td>
<td>SMS</td>
<td>+51</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Philippines</td>
<td>SMS</td>
<td>+63</td>
</tr>
<tr>
<td>Country</td>
<td>Supported Delivery Methods</td>
<td>Country Code</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Poland</td>
<td>SMS, Voice call</td>
<td>+48</td>
</tr>
<tr>
<td>Portugal</td>
<td>SMS, Voice call</td>
<td>+351</td>
</tr>
<tr>
<td>Puerto Rico</td>
<td>SMS, Voice call</td>
<td>+1</td>
</tr>
<tr>
<td>Russia (Russian Federation)</td>
<td>SMS, Voice call</td>
<td>+7</td>
</tr>
<tr>
<td>Rwanda</td>
<td>SMS</td>
<td>+250</td>
</tr>
<tr>
<td>Saint Barthélemy</td>
<td>SMS</td>
<td>+590</td>
</tr>
<tr>
<td>Saint Kitts and Nevis</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td>Saint Lucia</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td>Saint Martin (French side)</td>
<td>SMS</td>
<td>+590</td>
</tr>
<tr>
<td>Saint Pierre and Miquelon</td>
<td>SMS</td>
<td>+508</td>
</tr>
<tr>
<td>Saint Vincent and the Grenadines</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td>Samoa</td>
<td>SMS</td>
<td>+685</td>
</tr>
<tr>
<td>San Marino</td>
<td>SMS, Voice call</td>
<td>+378</td>
</tr>
<tr>
<td>São Tomé and Principé</td>
<td>SMS</td>
<td>+239</td>
</tr>
<tr>
<td>Saudi Arabia</td>
<td>SMS, Voice call</td>
<td>+966</td>
</tr>
<tr>
<td>Senegal</td>
<td>SMS</td>
<td>+221</td>
</tr>
<tr>
<td>Serbia</td>
<td>SMS, Voice call</td>
<td>+381</td>
</tr>
<tr>
<td>Country</td>
<td>Supported Delivery Methods</td>
<td>Country Code</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Seychelles</td>
<td>SMS</td>
<td>+248</td>
</tr>
<tr>
<td>Sierra Leone</td>
<td>SMS</td>
<td>+232</td>
</tr>
<tr>
<td>Singapore</td>
<td>SMS</td>
<td>+65</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Slovakia</td>
<td>SMS</td>
<td>+421</td>
</tr>
<tr>
<td>(Slovak Republic)</td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Slovenia</td>
<td>SMS</td>
<td>+386</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Solomon Islands</td>
<td>SMS</td>
<td>+677</td>
</tr>
<tr>
<td>Somalia</td>
<td>SMS</td>
<td>+252</td>
</tr>
<tr>
<td>South Africa</td>
<td>SMS</td>
<td>+27</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>South Korea</td>
<td>SMS</td>
<td>+82</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>South Sudan</td>
<td>SMS</td>
<td>+211</td>
</tr>
<tr>
<td>Spain</td>
<td>SMS</td>
<td>+34</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Sri Lanka</td>
<td>SMS</td>
<td>+94</td>
</tr>
<tr>
<td>Sudan</td>
<td>SMS</td>
<td>+249</td>
</tr>
<tr>
<td>Suriname</td>
<td>SMS</td>
<td>+597</td>
</tr>
<tr>
<td>Svalbard and Jan Mayen</td>
<td>SMS</td>
<td>+47</td>
</tr>
<tr>
<td>(Norway)</td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Swaziland</td>
<td>SMS</td>
<td>+268</td>
</tr>
<tr>
<td>Sweden</td>
<td>SMS</td>
<td>+46</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Switzerland</td>
<td>SMS</td>
<td>+41</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Syria</td>
<td>SMS</td>
<td>+963</td>
</tr>
<tr>
<td>(Syrian Arab Republic)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Taiwan</td>
<td>SMS</td>
<td>+886</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Tajikistan</td>
<td>SMS</td>
<td>+992</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Country</td>
<td>Supported Delivery Methods</td>
<td>Country Code</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Tanzania</td>
<td>SMS</td>
<td>+255</td>
</tr>
<tr>
<td>Thailand</td>
<td>SMS</td>
<td>+66</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Timor-Leste (East Timor)</td>
<td>SMS</td>
<td>+670</td>
</tr>
<tr>
<td>Togo</td>
<td>SMS</td>
<td>+228</td>
</tr>
<tr>
<td>Tonga</td>
<td>SMS</td>
<td>+676</td>
</tr>
<tr>
<td>Trinidad and Tobago</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area Code: 868</td>
</tr>
<tr>
<td>Tunisia</td>
<td>SMS</td>
<td>+216</td>
</tr>
<tr>
<td>Turkey</td>
<td>SMS</td>
<td>+90</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Turkish Republic of</td>
<td>SMS</td>
<td>+90</td>
</tr>
<tr>
<td>Northern Cyprus</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turkmenistan</td>
<td>SMS</td>
<td>+993</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Turks and Caicos Islands</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area Code: 649</td>
</tr>
<tr>
<td>Tuvalu</td>
<td>SMS</td>
<td>+688</td>
</tr>
<tr>
<td>U.S. Virgin Islands</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Area Code: 340</td>
</tr>
<tr>
<td>Uganda</td>
<td>SMS</td>
<td>+256</td>
</tr>
<tr>
<td>Ukraine</td>
<td>SMS</td>
<td>+380</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>United Arab Emirates</td>
<td>SMS</td>
<td>+971</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>United Kingdom</td>
<td>SMS</td>
<td>+44</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>United States</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Multiple Area Codes</td>
</tr>
<tr>
<td>Uruguay</td>
<td>SMS</td>
<td>+589</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Uzbekistan</td>
<td>SMS</td>
<td>+998</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td></td>
</tr>
<tr>
<td>Country</td>
<td>Supported Delivery Methods</td>
<td>Country Code</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Vanuatu</td>
<td>SMS</td>
<td>+678</td>
</tr>
<tr>
<td>Vatican City</td>
<td>SMS, Voice call</td>
<td>+379</td>
</tr>
<tr>
<td>Venezuela</td>
<td>SMS, Voice call</td>
<td>+58</td>
</tr>
<tr>
<td>Vietnam</td>
<td>SMS, Voice call</td>
<td>+84</td>
</tr>
<tr>
<td>Virgin Islands, British</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td>Area Code: 284</td>
</tr>
<tr>
<td>Virgin Islands, U.S.</td>
<td>SMS</td>
<td>+1</td>
</tr>
<tr>
<td></td>
<td>Voice call</td>
<td>Area Code: 340</td>
</tr>
<tr>
<td>Western Sahara</td>
<td>SMS</td>
<td>+212</td>
</tr>
<tr>
<td>Yemen</td>
<td>SMS</td>
<td>+967</td>
</tr>
<tr>
<td>Zambia</td>
<td>SMS</td>
<td>+260</td>
</tr>
<tr>
<td>Zimbabwe</td>
<td>SMS</td>
<td>+263</td>
</tr>
</tbody>
</table>
Device ID Authentication

Device ID Authentication allows account administrators to restrict login to only approved devices. Devices can be registered in NetSuite using a unique identifier. After reviewing registered devices, the account administrator can approve or reject individual devices. Only devices approved by the administrator can log in.

The Device ID feature is enabled by default. No special setup or configuration in NetSuite by account administrators is required to use the device record.

See the following for information on using this feature:

- Device ID and the SCIS SuiteApp
- Managing Devices on the List of devices Page
- The Device Record
- Creating Device Records Manually
- Viewing System Notes
- Deleting a Device Record

Device ID and the SCIS SuiteApp

Currently, the Device ID feature is intended for use with Suite Commerce InStore (SCIS) SuiteApp and the point-of-sale (POS) devices running the SCIS POS application.

For more information on SCIS, see the help topic SuiteCommerce InStore Administrator's Overview. See also, Installing SCIS Mobile Apps.

Device records are automatically created in NetSuite as users log in for the first time using POS devices with the SCIS POS application installed. Users are then notified that they must wait for the device to be approved before they can use SCIS on that device. NetSuite account administrators maintain a list of approved POS devices in NetSuite. Only the devices that have been reviewed and approved have access to SCIS.

When the SuiteCommerce InStore SuiteApp is installed in a NetSuite account, it automatically creates a role restricted by device ID. This is the only role allowed to log in to SCIS on a device configured with the SCIS POS application.

Users with device ID role who attempt to log in using an authorized device receives the error message "No device id role was found". The user must contact the account administrator and request to be assigned an SCIS device ID role.

Account administrators can create additional device ID restricted roles if desired, using the SCIS-created roles as a template. For more information on SCIS roles, see the help topic SCIS Roles and Permissions.

The first time a user attempts to log in to SCIS on a device running the SCIS POS application, a unique device identifier is sent to NetSuite. The name of the device is also sent. With this information, a Device record is created in NetSuite in Pending status. Users cannot log in to SCIS with the device until the NetSuite account administrator reviews the device record and changes the device status to Trusted. This requirement ensures that only devices approved by the NetSuite account administrator can be used to log in. The account administrator can also change the device status to block a device, or put a device record on hold.
Note: Users with device ID-restricted roles are not asked security questions. See the help topic Setting Up Security Questions for more information.

Managing Devices on the List of devices Page

With SCIS and the SCIS POS application, device records are automatically created in NetSuite as users log in from the POS devices for the first time. From the List of devices page, account administrators maintain and manage the list of the POS devices that can potentially access the SCIS website in your NetSuite account.

The List of devices page

To view the List of devices page, go to Setup > Integration > Device ID.

The following screenshot is an example of two device records that were created automatically when the users logged in from POS devices for the first time. (The SCIS POS application passed in the device ID and the device name when the user attempted to log in with the device.)

Both devices are in Pending status, and the device ID is displayed in full.

Important: To allow the account administrator the opportunity to verify the device, the device ID is displayed in full. As soon as the account administrator changes the status, the device ID is masked and cannot be retrieved from the system.

Note: After the device record has been created, NetSuite recommends you do not change the device name on the device. If the name is changed on the device, administrators would need to make the corresponding update to the device record in NetSuite. The device record in NetSuite is never updated by the POS application on the device after the initial login creates the record.

In the following screenshot, the account administrator has created two additional device records manually. (See Creating Device Records Manually for more information.)

In this example, the administrator did not provide the optional Device Name when creating the records, and changed the device status to Pending before saving each record. NetSuite automatically created a Device Name using the Device ID, masking everything except the last four digits.
The following screenshot is an example of the account administrator reviewing the List of devices, and changing the status of each device. The account administrator is sure the Device IDs for the automatically created records are correct, and changes the status to Trusted.

For the manually created records, the account administrator wants more time to verify the Device ID numbers are correct, and changes the status for these records to On Hold. The following screenshot shows the list of devices after the statuses were changed, and the page was refreshed. All of the Device IDs have been masked, except for the last four digits.

**Important:** The only time the Device IDs are shown in full is when a device remains in the status in which it was initially created. This allows the account administrator to verify the Device ID. Treat Device IDs as securely as you would treat a password.

As soon as the device status is changed, the Device ID is masked, and cannot be retrieved from the system.

The Device Record

Account administrators can review the list of device records in their NetSuite account. Go to Setup > Integration > Device ID to access the List of devices page.

- To open a record, click Edit or View in the appropriate row on the List of devices page.
- To create a device record manually, click New Device ID. See Creating Device Records Manually.

**Note:** An alternative method of creating a new device manually is available on the Device page. Select **New** from the **Actions** list to open a blank Device record.

- To view the System Notes for a device, see Viewing System Notes
- To delete a device record, see Deleting a Device Record.
Creating Device Records Manually

It is possible for account administrators to create device records manually, but this approach is not recommended due to the potential for data entry errors. Before creating the record, ensure you have access to the correct device ID and device name.

To create a device record:

1. Go to Setup > Integration > Device ID.
2. Click **New Device ID**. The **Device** tab is displayed by default.
3. Enter the **Device Name**. If you do not enter a device name, a name will be generated automatically.
4. Enter the **Device ID**. The device ID should be a unique identifier for a specific device.
5. Manually created records default to the **Device Status** of Trusted, meaning the device is allowed to log in to NetSuite. Change the status, if desired. Devices in any status other than Trusted will not be allowed to log in to NetSuite.
   - **Pending**: Awaiting review and approval from an account administrator. For device records created manually, NetSuite recommends changing the status to Pending before saving the record. This allows additional time to verify the information (especially the Device ID) has been entered correctly.
   - **On Hold**: Has been reviewed by the account administrator, but is not yet approved. This status could be used for devices for a store that is not yet open, for example.
   - **Denied**: Reviewed by the account administrator and denied access to NetSuite.
6. Click **Submit**.

Viewing System Notes

Account administrators can view the change history of a device record on the System Notes tab.

To view the system notes for a device:

1. Go to Setup > Integration > Device ID.
2. On the List of devices page, click **Edit** or **View** for a particular device.
3. On the Device page, click the **System Notes** tab.
Deleting a Device Record

Account administrators can delete a device record. For example, you may want to delete a device that was created manually if information such as the device ID was not entered correctly.

To delete a device record:

1. Go to Setup > Integration > Device ID.
2. On the List of devices page, click **Edit** or **View** for a particular device.
3. Select **Delete** from the **Actions** list.
4. If you are sure you want to delete this device, click **OK**.

A confirmation notice displays on the List of devices page.

**Note:** You can search for a device's login history using the Login Audit Trail search capabilities, even after the device record has been deleted. See the help topic [Login Audit Trail Overview](#). Only a device that has been used for login leaves a login audit trail. Searching for a device that was never used for login will have no results.
Outbound Single Sign-on (SuiteSignOn)

The outbound single sign-on implementation in NetSuite is called SuiteSignOn. SuiteSignOn enables users to be authenticated in the NetSuite user interface. Then, users can move directly from a link in the NetSuite UI to an external user-authenticating web application, without supplying additional authentication. These links, called connection points, currently are supported in NetSuite custom subtabs, custom portlets, Suitelets and user event scripts. NetSuite provides a SuiteSignOn setup page where application providers can enter data used for connection points.

**Note:** Calls initiated by SOAP web services are not supported by SuiteSignOn.

**Note:** SuiteSignOn access from your web store is supported. After reading through the Outbound SSO help topics, see also Outbound Single Sign-on (SuiteSignOn) Access from Your Web Store.

**SuiteSignOn Benefits**

The SuiteSignOn feature provides the following benefits:

- **Improved usability:** Users can access other applications with their NetSuite login credentials, so they can complete daily tasks more quickly. They do not need to repeatedly log in and log out of multiple applications, or manage multiple sets of login credentials. They can log in a single time to NetSuite, and access an integrated solution within a single user interface.

- **Increased security and central access control:** The password policy that is enforced for NetSuite access is enforced for any integrated application, providing consistency and limiting potential security issues.

- **Reduced IT and support costs:** The rollout of integrated applications is simplified because there is no need to maintain multiple databases for user credentials and access control.

- **NetSuite as the single trusted system for authentication:** Access from the NetSuite user interface to an external application user interface is confined to an iFrame. The external application does not have rights to change data in NetSuite except through specialized SOAP web services calls.
More secure SOAP web services integrations: The integrated application can use an already active session to transmit data to NetSuite through SOAP web services calls, instead of requiring the user to log in again. Changes submitted through SOAP web services are reflected in the NetSuite audit trail for the logged in user who makes the specific changes. SOAP web services use the same role that was used to log in the user to NetSuite.

Important: If you are attempting to implement inbound single sign-on from an external application to NetSuite, use one of the following NetSuite inbound SSO features:

- SAML Single Sign-on
- OpenID Single Sign-on
- Inbound Single Sign-on

See also Authentication Overview, which includes a Single Sign-on (SSO) Overview section.

SuiteSignOn Overview

SuiteSignOn provides seamless integration of NetSuite with other applications through outbound single sign-on. With this feature, NetSuite users can access external applications directly from the NetSuite user interface without additional authentication.

Anyone using the SuiteSignOn feature should see the following topics:

- Outbound Single Sign-on (SuiteSignOn)
- SuiteSignOn Sequence Diagram and Connection Details
- Understanding SuiteSignOn
- SuiteSignOn Required Features (A SuiteSignOn solution cannot be implemented until certain features are enabled.)

Application providers who want to sell their applications and services to customers who also use NetSuite must see these topics as well:

- Setting Up SuiteSignOn Integration
- Creating a SuiteSignOn Bundle
- SuiteSignOn Definitions, Parameters, and Code Samples

Application developers who may need more information about certain NetSuite features for creating a SuiteSignOn solution should see the following topics:

- For scripting, see the help topic SuiteScript 2.0.
- For SOAP web services, see the help topic SuiteTalk SOAP Web Services Platform Overview.
- For SuiteApps (bundles), see the help topic SuiteBundler Overview.
- See also Troubleshooting SuiteSignOn (Outbound SSO).

Administrators who will be responsible for exposing third-party applications to NetSuite users should see Making SuiteSignOn Integrations Available to Users. In cases where the administrator might also be responsible for building a custom integration should see the topics pertaining to application providers.
**Important:** Be aware of the following: Administrators should exercise caution when integrating with third-party applications using SuiteSignOn. Some integrations may require access to, or even modify, some data in your NetSuite account. Make sure you review the data requirements and understand what kind of information is accessed, retrieved, modified, or deleted by the third-party system. NetSuite has no control, responsibility, or liability regarding any third-party applications, even if NetSuite offers resale and integration options for customers' convenience. You use and integrate with third-party applications at your sole risk.

**Understanding SuiteSignOn**

Each location in the NetSuite user interface where users can access an external application through SuiteSignOn is called a connection point. An application can have multiple connection points on different NetSuite pages. Currently, custom subtabs, custom portlets, and Suitelets are supported as connection points. User event scripts also are supported as connection points for SOAP web services integrations between NetSuite and external applications.

To implement single sign-on integration with an application, the application provider needs to set up information for each connection point. This information includes the name of the NetSuite subtab, portlet, Suitelet, or user event script connection point, the external application landing page, optional data that sets context for the landing page, and optional user identification data.

NetSuite authenticates each user upon login. When a user accesses a connection point, NetSuite initiates a two-way communication with the external application, and verification data passes between the two applications. This communication is referred to as a handshake, because of its back and forth nature. After the handshake has been completed, the external application landing page displays in the subtab, portlet, or Suitelet interface. Also, if the application provider has included related SOAP web services calls in their application code, users' edits to external application data can be transferred to NetSuite.

For more information about how SuiteSignOn connections work, see SuiteSignOn Sequence Diagram and Connection Details.

For an overview of the ways in which a company can benefit from a SuiteSignOn implementation, see SuiteSignOn Benefits.

**SuiteSignOn Sequence Diagram and Connection Details**

See the following sections for information about SuiteSignOn.

- SuiteSignOn Sequence Diagram
- SuiteSignOn Connection Details

**SuiteSignOn Sequence Diagram**

The following sequence diagram illustrates the interaction between NetSuite and an external application during a SuiteSignOn connection.

- Steps 1 and 2 occur in the NetSuite user Interface.
- Steps 3-6 represent the handshake, meaning the calls required to verify the user and display the application in the NetSuite user interface.
Steps 8-9 represent optional SOAP web services calls, used if the application provider wants to enable data transfer from the external application to NetSuite.

A detailed description of each step follows the sequence diagram.

### SuiteSignOn Connection Details

See the following detailed steps for each action shown in the preceding SuiteSignOn connection sequence diagram.

1. **User logs in to NetSuite, initiating a NetSuite session.**
2. **User clicks on one of the following in the NetSuite user interface:**
   - A subtab that provides SuiteSignOn access
   - A page displaying a portlet that provides SuiteSignOn access
   - A link for a Suitelet that provides SuiteSignOn access
   - An action button that results in the execution of a user event script that provides SuiteSignOn access
3. **Outbound single sign-on request**: NetSuite generates a token, and sends this token to the external application as the value for the `oauth_token` URL parameter. This outbound HTTP call also includes a `dc` and an `env` URL parameter. These values can be mapped to the URL to be used for NetSuite access (see Mappings of dc and env URL Parameter Values and Understanding NetSuite URLs). If any data fields were previously defined as required context for the connection, NetSuite sends values for these fields at the same time.
4. **Verify request**: The external application sends back to NetSuite the token, the consumer key, and the signature, along with other information such as the timestamp and nonce, to verify the user.

   The consumer key is a unique identifier for the application provider, generated by NetSuite when the application provider sets up a SuiteSignOn connection. The signature is computed from the shared secret, the password defined by the application provider during this setup, based on the
OAuth 1.0 standard. For information on computing the signature, see Generate the Signature for the OAuth Header for Outbound SSO for information about the signature. See also the OAuth 1.0 Protocol, RFC 5849.

5. **Verify response**: NetSuite responds to the verification, sending any user identification information that was previously defined as necessary for the connection, in XML format. This information will be used by external application to uniquely identify the NetSuite user. For details about secure combinations of fields that should be used to uniquely identify users, please read Choosing User Identification Fields for SuiteSignOn.

6. **Outbound single sign-on response**: The external application sends the HTML for the landing page, and the page displays. Or, if there is a problem, an error is returned instead.

   **Note**: These steps may or may not occur, depending on the situation:
   - For user event script connection points, step 6 is omitted.
   - Steps 7 and 8 are optional. If a SOAP web services request is sent (step 8) then NetSuite sends a SOAP web services response (step 9).

7. The user makes changes in the external application page displayed in NetSuite, then saves them.

8. **SOAP web services request**: The external application sends a SOAP web services request, that includes the token and shared secret along with other verification data, to NetSuite.

9. **SOAP web services response**: NetSuite sends a SOAP web services response to the external application, and either the changes are saved to NetSuite, or an error is returned. SOAP web services uses the same role that was used to log in to NetSuite.

   **Note**: Be aware of the following:
   - The token that NetSuite generates is good for the length of the UI session, or for 20 minutes of inactivity.
   - If a user repeats step 2 multiple times during a single session, steps 4-5 can be skipped (at the discretion of the third-party client) after the first time.
   - If the user logs out of NetSuite and logs back in, or switches roles, when the user clicks on the connection point, a new token is generated.

---

**SuiteSignOn Required Features**

Application providers and NetSuite administrators may need to enable certain features to facilitate a SuiteSignOn implementation. The following table lists each feature and whether it must be enabled, depending on the scenario.

All of the features below can be enabled at Setup > Company > Enable Features, on the SuiteCloud subtab.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Required for Application Providers?</th>
<th>Required for SuiteSignOn User Accounts?</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuiteSignOn</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>SuiteTalk (web services)</td>
<td>Maybe</td>
<td>Maybe</td>
<td>Required if SuiteSignOn code includes SOAP web services calls.</td>
</tr>
<tr>
<td>SuiteBundler</td>
<td>Yes</td>
<td>No</td>
<td>Required to create and deploy bundles. Not required to install bundles.</td>
</tr>
</tbody>
</table>
### SuiteSignOn Required Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Required for Application Providers?</th>
<th>Required for SuiteSignOn User Accounts?</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client and Server SuiteScript</td>
<td>Maybe</td>
<td>Maybe</td>
<td>Required if the connection points are created using custom portlets, Suitelets, or user event scripts, client and server SuiteScript should be enabled. Not required for subtab connection points.</td>
</tr>
</tbody>
</table>

### Setting Up SuiteSignOn Integration

The following tasks can be completed by anyone who has the SuiteSignOn permission. However, most of these tasks will be completed by application providers wanting to implement a SuiteSignOn integration with NetSuite.

If you are a NetSuite administrator who is working with an application provider to build a custom solution (or you are managing a bundled solution), you may also end up performing some of these tasks. Typically, however, most administrators will complete the tasks outlined in Making SuiteSignOn Integrations Available to Users.

#### Summary of integration tasks:

2. Application providers must add required code to their application to support the exchange of token and shared secret information with NetSuite, referred to as the handshake. For sample code, see SuiteSignOn Definitions, Parameters, and Code Samples. These code additions include:
   - a verify call in the HTTP header and code that requests token verification from NetSuite
   - (optional) SOAP web services calls to transfer data between NetSuite and your application
3. Create one or more custom subtabs, portlets, Suitelets or user event scripts to be connection points that provide access to the integrated application. See Creating SuiteSignOn Connection Points.

**Important:** Only a Suitelet connection point is supported for SuiteSignOn access from your web store.

4. (Optional) Define any custom entity fields as user identification fields.
   a. Ensure that these fields have been created, and that the Available to SuiteSignOn box is checked.

   **Important:** Do not check the Use Encrypted Format box.
   b. You must determine a way for account administrators to enter or import values for these fields as needed.

   See Using Custom Fields as SuiteSignOn User Identification.
6. (Application providers) Create a SuiteBundle that includes SuiteSignOn connection data and custom objects, write bundle documentation instructing administrators how to set it up in their accounts, and make the bundle available to NetSuite users. See Creating a SuiteSignOn Bundle.
7. (NetSuite administrator) Install the SuiteSignOn bundle created by the application provider. See Making SuiteSignOn Integrations Available to Users.
Creating SuiteSignOn Records

You must create one SuiteSignOn record for each application that you want to integrate with NetSuite. Each application may have multiple connection points, all of which are listed on the same record. You must create a separate SuiteSignOn record in each account type where you want to use SuiteSignOn. For example, for each application, create a record in your production account, and then create a separate record if needed in your Release Preview or sandbox account. The records cannot be shared between accounts, because the accounts do not have the same account ID.

**Important:** You must have the SuiteSignOn permission to create or edit SuiteSignOn records.

- To create a new SuiteSignOn record for an application, go to Setup > Integration > SuiteSignOn > New.
- To edit an existing SuiteSignOn record for an application, go to Setup > Integration > SuiteSignOn and click the Edit link for a record.
  
  See Editing SuiteSignOn Records.

On the SuiteSignOn page, you can complete the following tasks:

- Setting SuiteSignOn Basic Definitions
- Defining SuiteSignOn Connection Points
- Choosing User Identification Fields for SuiteSignOn (optional)
- Using Custom Fields as SuiteSignOn User Identification (optional)
- Dynamically Mapping User Identification Information (optional)

After you are done with these tasks, you can build a SuiteSignOn bundle to distribute to your customers. See Creating a SuiteSignOn Bundle.

**Warning:** For bundled SuiteSignOn integrations, the SuiteSignOn record is completed by the application provider, and administrators install the bundle that contains this data, making edits to the record as instructed in the bundle document. If administrators attempt to make other edits to this page, issues are likely to arise. See Making SuiteSignOn Integrations Available to Users.

Setting SuiteSignOn Basic Definitions

On the SuiteSignOn record for an external application, you (the application provider) must define the following:

- **Name** - A name for the external application integration, to display in NetSuite lists.
- **ID** - A script ID for this integration, to be passed as a parameter in the portlet script. The value you enter here is automatically prepended with `customsso`. You should assign a unique script ID to your SuiteSignOn object if you intend to bundle and distribute your integration.
- **Shared Secret** - A password used to establish ownership of the Consumer Key generated by NetSuite. This value is included in the signature passed in your HTTP header, and needs to be referenced in your application verification code.

**Important:** See Notes about Modifying the Shared Secret for tips about changes to this password.

You do not need to define the following:
Creating SuiteSignOn Records

- **Consumer Key** - You cannot enter or edit this globally unique identifier for your application. It is generated by NetSuite. You must include this value in your HTTP header and application verification code.

- **Partner Account** - Each customer's account ID for your application. Each customer may need to enter this value after installation of the SuiteSignOn bundle. This value is not necessary if your integrated application does not require this value for identification. Be sure to include instructions for this task in your bundle documentation, if necessary.

- **Web Services Access** - Level of access supported for SOAP web services callbacks from integrated applications. The following options are available:
  - **Same as UI Role** - the default, which allows SOAP web services callbacks from integrated applications with the same level of permissions as in the user interface integration.
  - **No Access** - prevents integrated applications from accessing NetSuite through SOAP web services callbacks.
  - Additional options for any roles designated as **Web Services Only** in the account. Selecting one of these roles allows SOAP web services callbacks from integrated applications, but limits access to the permissions levels assigned to the selected role.

As a security best practice, you should provide the minimum level of access required for SuiteSignOn integrated applications. For example, if an application only requires user interface integration, it is best to set the **Web Services Access** option to **No Access**.

The **Web Services Access** field is also available for viewing and editing on the SuiteSignOn list page at Setup > Integration > SuiteSignOn.

After you have set basic definitions for the SuiteSignOn integration, you can define one or more connection points where your application is displayed in the NetSuite user interface, and user identification fields that are used as context for the integration.

**Note:** For examples of HTTP header and application verification code, see SuiteSignOn Definitions, Parameters, and Code Samples.

## Defining SuiteSignOn Connection Points

Connection points are the locations in the NetSuite user interface that provide access to your application. Every application integrated through SuiteSignOn can have multiple connection points.

To set up connection points for your application, define the following on the SuiteSignOn page:

- **URL** - Enter the URL for the external application landing page to be displayed in the connection point. This page must be secure, with an https:// URL. SuiteSignOn is not supported for http:// sites. You can specify a different URL for each connection point.

- **Integration Variables** - You can optionally define one or more field values to be passed as context in the initial HTTP call from NetSuite to your application, before authentication. For an example of this call, see NetSuite HTTP Outbound Call.
  - You do not need to specify integration variables if context is included in the URL.
  - Both static and dynamic field values are supported.
    - To specify a static value, use a format like `q=value`, as in the following examples:
      ```
customer_id=12345
first=John
last=Smith
mid=Jay
```
To specify a dynamic value, use a format like $q=$field_id$, as in the following examples:

- customer_id = $id$
- first = $firstname$
- last = $lastname$
- mid = $middlename$

To specify a null value, use a format like $q=\square$

- Variables can include spaces. Static variables cannot include commas within values.
- You can use comma-separated values or carriage return-separated values to specify multiple values.
- You cannot use social security numbers, passwords, or credit card numbers as integration variables, because of the potential security risks. If you do so, they will not be returned.
- For subtab connection points, standard and custom fields on the form for the specified record type can be used as integration variables. You cannot use fields that are not included on the form. If a referenced field has no value, no value is passed as an integration variable.
- For portlet, Suitelet, and user event connection points, see Defining Integration Variables for Connection Points. This section provides detailed information regarding the use of static and dynamic integration variables in these connection points.

- Display Type - Choose Subtab, Portlet, Suitelet, or User Event.
- Display Context - Choose the name of the subtab, portlet, Suitelet, or user event script to be used for SuiteSignOn access.

A subtab, portlet script, Suitelet, or user event script must already exist in your account to be included in the dropdown list. See Creating a Custom Subtab Connection Point, Creating a Portlet Connection Point, Creating a Suitelet Connection Point, and Creating a User Event Connection Point.

- Record Type - (Subtabs only) Choose the record type for each subtab connection point. Custom record types are available for their associated subtabs.

If you want to display the external application in a custom subtab on multiple record types (for example, on contacts, customers, and partners), you must add multiple connection points, with the same Display Type and Display Context, and a different Record Type for each.

Note: Be sure to click Add after you enter each connection point.

Defining Integration Variables for Connection Points

You can define both static and dynamic integration variables for portlet, Suitelet, and user event connection points.

Note: If you want to define dynamic integration variables for either of these connection types, you must do so in the portlet, Suitelet, or user event JavaScript file. You cannot define dynamic integration variables in the Integration Variables field on the SuiteSignOn record.

The following code sample shows a portlet script that is used to get the email address of the currently logged-in NetSuite user. The value of the email address can then be passed to the URL as a dynamic integration variable.

```javascript
/**
 * @NApiVersion 2.x
 * @NScriptType Portlet
 */
```
Creating SuiteSignOn Records

The following screenshot shows that you could have used the Integration Variables field on the SuiteSignOn record to define the `partner` integration variable, which is static. You could not have used the Integration Variables field to define a dynamic integration variable, such as the `email` variable in the preceding script.

Choosing User Identification Fields for SuiteSignOn

The user identification fields are used by external applications to uniquely identify the NetSuite user. User identification fields are defined per application, so they are the same for each connection point. These fields' values are passed in XML format in the HTTP response from NetSuite to your application after verification. For an example, see NetSuite HTTP Verify Call Response.

The following user identification fields are provided on the SuiteSignOn page:

- Email - Email address used as the user ID for NetSuite
- Account - Customer's NetSuite account ID
- First Name
- Middle Name
- Last Name
- Internal ID - NetSuite-generated unique identifier
- External ID - External application unique identifier stored in NetSuite

To uniquely identify each user across all NetSuite accounts, you should use one of the following two combinations of data:

- The customer's NetSuite account ID and the user's email address.
- The customer's NetSuite account ID and the user's internal ID.

You also can make custom entity fields available on the SuiteSignOn record, by checking the Available to SuiteSignOn box on the Custom Entity Field record. See Using Custom Fields as SuiteSignOn User Identification.
Creating SuiteSignOn Records

87

Using Custom Fields as SuiteSignOn User Identification

In addition to the standard fields available when you are Choosing User Identification Fields for SuiteSignOn, you can define entity custom fields to be available for this purpose. NetSuite passes values of the user identification fields selected on the SuiteSignOn page to your application.

**Important:** In addition to any custom fields that you choose to include, every user should be identified by a unique combination of data defined in standard NetSuite fields. For details on the two combinations that are supported, see Choosing User Identification Fields for SuiteSignOn.

**To make a custom entity field available for user identification:**

1. If the field does not yet exist in NetSuite, add it at Customization > Lists, Records, & Fields > Entity Fields.
2. Ensure that the field is available on all types of records corresponding to the users who can access your SuiteSignOn integration. These may include employees, customers, partners, and vendors.
3. Check the Available to SuiteSignOn box. After this box is checked, the custom field is listed on the User Identification subtab of the SuiteSignOn page.

**Important:** Do not check the Use Encrypted Format box.

For instructions for setting up custom fields, see the help topic Creating a Custom Field.

You must determine a way for account administrators to populate custom field values, either through manual entry, mass update, CSV import, or another import process. You should provide instructions for this task in your SuiteSignOn bundle documentation. If you want values to be populated through CSV import, you can save an import map and include it in the bundle. See the help topics Working with Saved CSV Imports and Creating a SuiteSignOn Bundle.

Dynamically Mapping User Identification Information

If the default identity information returned by standard SuiteSignOn ID fields is insufficient for a certain application, you can establish dynamic identity mappings. To establish dynamic identity mappings, you can design a landing page and prompt each user upon first connection to log in. Alternatively, enter their ID in the third-party system, and submit the mapping back into a custom entity field, which is available to SuiteSignOn through SOAP web services.

After the field is populated, subsequent connections into the third-party application will bypass the initial landing page and the identity of the user is known. For this approach to work, the role of the logged-in user in NetSuite should have permission to update their own entity record to set the custom field.
If the logged in user’s role does not have permission to update the entity record, a custom record can be created to track identity mappings for a certain SuiteSignOn integration. Another entity custom field that is available to SuiteSignOn can source the mapping from the custom record.

For instructions for setting up custom fields, in the NetSuite Help Center see the help topic Creating a Custom Field. For instruction on creating custom records, see the help topic Custom Records.

Creating SuiteSignOn Connection Points

A connection point is the place in the NetSuite user interface where users access the external application. A single application can have multiple connection points on different NetSuite pages.

Currently, custom subtabs, custom portlets, Suitelets, and user event scripts are supported as connection points.

❗ Note: Calls initiated by SOAP web services are not supported by SuiteSignOn.

The type of connection point you create depends on how you want NetSuite users to access the integrated application.

See Comparing Subtab, Portlet, Suitelet and User Event Connection Points to help you decide which type of connection point best suits your implementation.

To create one of the supported connection points, see these topics:

- Creating a Custom Subtab Connection Point
- Creating a Portlet Connection Point
- Creating a Suitelet Connection Point
- Creating a User Event Connection Point

❗ Important: Only a Suitelet connection point is supported for SuiteSignOn access from your web store.

Comparing Subtab, Portlet, Suitelet and User Event Connection Points

If you are trying to decide whether to set up subtab, portlet, Suitelet, or user event connection points for your application, consider the following: how comfortable you are with scripting, where the application should display within the NetSuite user interface, and how you want it to look.

- Subtab connection points may be simplest to implement, because they do not require scripting.
- A portlet connection point provides greater flexibility than a subtab in how the application looks. A Suitelet provides even more flexibility.
- A Suitelet is the only connection point supported for SuiteSignOn access from your web store.
- A user event connection point provides integration with an external application without exposing it in the NetSuite user interface.

The following table outlines differences:
Creating SuiteSignOn Connection Points

### Functionality

<table>
<thead>
<tr>
<th>Required NetSuite customizations</th>
<th>Subtab</th>
<th>Portlet</th>
<th>Suitelet</th>
<th>User Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creation of a custom subtab. No scripting required.</td>
<td>Custom scripting required.</td>
<td>Custom scripting required.</td>
<td>Custom scripting required.</td>
<td></td>
</tr>
</tbody>
</table>

### Availability on dashboard

| | Visible within specifically defined records. | Can be added to any page that allows custom portlets. | Link can be added to any page menu. | Not exposed in user interface. Connection is initiated either Before Load, Before Submit, or After Submit, based on the user event script record function. |
| | Automatically available after bundle installed. | Not available until custom portlet is added and configured to display script. | Automatically available after bundle installed. | |

### Ability to modify application “look and feel”

| | Very limited. External application landing page displayed within iFrame is only subtab contents. | Based on script, so can be free-form. | Based on script, so can be free-form. | Not exposed in user interface. |

### Required connection point definitions

| | Must define a separate connection point for each subtab integration. | One connection point can provide integration in multiple portlets. | Must define a separate connection point for each Suitelet integration. | Must define a separate connection point for each integration. |

### Creating a Custom Subtab Connection Point

You can create custom subtabs to provide outbound single sign-on access within NetSuite records. The following types of custom subtabs are available:

- **Transaction** — Can be displayed on transaction records such as sales order, cash sale, opportunity
- **Entity** — Can be displayed on entity records such as customer, vendor, employee
- **Item** — Can be displayed on item records such as inventory, non-inventory, assembly/bill of materials
- **CRM** — Can be displayed on CRM records such as task, phone call, event
- **Custom Record Subtab** — Can be displayed on its associated custom record

**Note:** For a complete list of transaction, entity, item, and CRM records that support custom subtabs, see the help topic Creating Custom Subtabs in the NetSuite Help Center.

### To create custom subtabs:

1. Go to Customization > Forms > Subtabs, and choose an option:
   - Click the subtab for the type of record where you want to create a new subtab: **Transaction**, **Entity**, **Item**, or **CRM**.
   - OR:
     - Edit a custom record type record, either by going to Customization > Lists, Records, & Fields > Record Types and clicking an **Edit** column link, or by going to Customization > Lists, Records, & Fields > Record Types > New, and click the **Subtabs** subtab.
2. Enter the name for your subtab in the **Title** field. You should use a name that is a meaningful reference to your application, because it serves as the subtab label in the NetSuite user interface.
3. If desired, designate this subtab as a child of an existing subtab. In the **Parent** field, select an existing subtab from the list.
4. Click Add.
5. Repeat these steps for each subtab you want to create.
6. Click Save.

After you have created a custom subtab, you can define it as a connection point on the SuiteSignOn page. For each subtab connection point, you can specify a single record type to which it applies. The subtab is displayed on that record type's forms. When the record that includes the subtab is loaded, the SuiteSignOn connection is initiated, resulting in the display of an iFrame rendering your application.

**Note:** Be aware of the following:
- Usually, you must add at least one custom field to a custom subtab for it to display on forms. However, subtabs that are defined as SuiteSignOn connection points do not require any custom fields. You should not add any fields to these subtabs.
- You can control the records to which a subtab is applied when you set up subtab connection points on the SuiteSignOn page. See Creating SuiteSignOn Records.
- You can limit the users who have access to each record type subtab by customizing the record type form and setting up preferred forms for users.
- For a description of the differences between different types of connection points, see Comparing Subtab, Portlet, Suitelet and User Event Connection Points.

### Creating a Portlet Connection Point

You can create portlet scripts to provide outbound single sign-on access in custom portlets. To make a portlet script available for a connection point, you must create a JavaScript file, create a NetSuite script record, and deploy the script.

For more information, see the following topics in the NetSuite Help Center:
- Portlet Scripts
- Running SuiteScript 1.0 in NetSuite Overview
- nlapiOutboundSSO(id)

**To create and deploy a SuiteSignOn portlet script:**

1. Create a .js file that uses SuiteScript API. For information about SuiteScript 1.0 API, see the help topic nlapiOutboundSSO(id). For information about SuiteScript 2.0 API, see the help topic sso.generateSuiteSignOnToken(options).
   For information about specifying values on the SuiteSignOn record, see Setting SuiteSignOn Basic Definitions.
2. Create a record for the script and deploy it in NetSuite. See the help topic Running SuiteScript 1.0 in NetSuite Overview.
3. Define a portlet connection point for your SuiteSignOn integration. See Defining SuiteSignOn Connection Points.

### Creating a Suitelet Connection Point

You can create Suitelets to provide outbound single sign-on access in custom user interface objects. To make a Suitelet available for a SuiteSignOn connection point, you must create a JavaScript file, create a NetSuite script record, and deploy the script.
Creating SuiteSignOn Connection Points

**To create and deploy a SuiteSignOn Suitelet:**

1. Create a `.js` file that uses SuiteScript API.
   - For information about SuiteScript 1.0 API, see the help topic `nlapiOutboundSSO(id)`.
   - For information about SuiteScript 2.0 API, see the help topic `sso.generateSuiteSignOnToken(options)`.
   - For information about specifying values on the SuiteSignOn record, see Setting SuiteSignOn Basic Definitions.
2. Create a record for the script and deploy it in NetSuite. See the help topic Running SuiteScript 1.0 in NetSuite Overview.
3. Define a Suitelet connection point for your SuiteSignOn integration. See Defining SuiteSignOn Connection Points.

Creating a User Event Connection Point

You can create user event scripts that use SuiteSignOn to support real-time integration between NetSuite and external applications. User event scripts execute at one of the following points: when a read operation on a record takes place (Before Load), when a record is submitted before changes are committed to the database (Before Submit), or when changes are committed to the database (After Submit). Through SuiteSignOn, a user event script can notify an external application of record updates, passing each record ID as a URL parameter. The external application can then access NetSuite through SOAP web services calls, to acquire additional information about these records.

**Important:** The external system’s access to NetSuite is limited to the access available for the user who performed the action that caused user event script execution.

To make a user event script available for a SuiteSignOn connection point, you must create a JavaScript file, create a NetSuite script record, and deploy the script.

**To create and deploy a SuiteSignOn user event script:**

1. Create a `.js` file that uses SuiteScript API.
   - For information about SuiteScript 1.0 API, see the help topic `nlapiOutboundSSO(id)`.
   - For information about SuiteScript 2.0 API, see the help topic `sso.generateSuiteSignOnToken(options)`.
   - For information about specifying values on the SuiteSignOn record, see Setting SuiteSignOn Basic Definitions.
2. Create a record for the script and deploy it in NetSuite. See the help topic Running SuiteScript 1.0 in NetSuite Overview.
3. Define a user event connection point for your SuiteSignOn integration. See Defining SuiteSignOn Connection Points.

Editing SuiteSignOn Records

After a SuiteSignOn record has been created, users with the SuiteSignOn permission can edit this record as needed.
If you make changes to a SuiteSignOn record after it has been bundled and distributed, you must update the bundle in the repository if it was copied there, and inform bundle users of the change, so they can update their installations to get the latest version. See the help topic Using the Bundle Repository.

**To edit a SuiteSignOn record:**

1. Go to Setup > Integration > SuiteSignOn, and click the **Edit** link for a record.
2. Make changes as desired. For details about definitions that can be edited, see:
   - Setting SuiteSignOn Basic Definitions
   - Defining SuiteSignOn Connection Points
   - Choosing User Identification Fields for SuiteSignOn
   - Using Custom Fields as SuiteSignOn User Identification

**Notes about Modifying the Shared Secret**

- You cannot change the Shared Secret value unless you are the creator of the SuiteSignOn record.
- If you change the Shared Secret after your SuiteSignOn solution has been installed in other accounts, you cause this password to change for all instances of the SuiteSignOn integration across all accounts in both the production and sandbox domains. So, for example, if you modify the Shared Secret on a SuiteSignOn record in a sandbox account, it is changed in production accounts as well.
- See Additional Shared Secret Requirements If Using PLAINTEXT for more information about requirements for the Shared Secret.

**Disabling a SuiteSignOn Integration**

You can mark a SuiteSignOn integration as inactive either on the record itself, by checking the Inactive box, or on the SuiteSignOn list page, by checking the Show Inactives box, then checking the Inactive box for the record. When a SuiteSignOn record is inactive, any subtab connection points are not displayed, and portlet scripts and Suitelets return errors.

**Creating a SuiteSignOn Bundle**

After you have completed the tasks in Setting Up SuiteSignOn Integration, you can use SuiteBundler to package your SuiteSignOn objects for distribution.

The following table lists common SuiteSignOn bundle objects:

<table>
<thead>
<tr>
<th>Object</th>
<th>When to include in SuiteSignOn Bundle</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuiteSignOn Outbound</td>
<td>Always</td>
</tr>
<tr>
<td>Connection</td>
<td>Any custom subtabs, portlet scripts, and Suitelets defined as connection points, and any custom fields defined as user identification, are automatically included with the SuiteSignOn Outbound Connection object.</td>
</tr>
<tr>
<td>Custom Field(s)</td>
<td>If integration uses custom fields as integration variables</td>
</tr>
<tr>
<td></td>
<td>Custom fields defined as user identification are automatically added.</td>
</tr>
</tbody>
</table>
Creating a SuiteSignOn Bundle

<table>
<thead>
<tr>
<th>Object</th>
<th>When to include in SuiteSignOn Bundle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saved CSV Import</td>
<td>If integration uses custom fields as integration variables or user identification, and you want to provide a predefined import mapping for populating these fields' values</td>
</tr>
</tbody>
</table>

You also should include bundle documentation, a file that provides instructions for account administrators who install the bundle.

SuiteSignOn bundles are customization bundles, not configuration bundles. For more information, see the help topic SuiteApp Creation and Distribution.

To bundle your SuiteSignOn integration:

1. Create your bundle documentation file. This file should include a description of the bundle contents and a list of steps that are required after bundle installation.
   - To help administrators verify SuiteSignOn page contents, you may want to include a checklist of values for the basic information fields, connection point details, and boxes that should be checked on the User Identification subtab. Or, you could include a screenshot of this page in the bundle documentation file.
   - For details about other steps you may need to explain in this file, see Making SuiteSignOn Integrations Available to Users.
   - You must include in the bundle any custom role or roles that are used for SOAP web services calls.

2. Go to Customization > SuiteBundler > Create Bundle to start the Bundle Builder, and follow the instructions in the SuiteBundler help topic Creating a Bundle with the Bundle Builder.

3. To enable administrators to install your bundle, communicate to them the bundle name and ID. Also let them know whether they should install the bundle from the bundle repository, from your production account, or from your sandbox account, and provide the account ID as necessary.

   **Note:** The Web Services Access option in a bundled SuiteSignOn record is pushed to target accounts as part of new bundle installations. However, a change to this option is not pushed to target accounts during bundle updates, to prevent overwriting account administrators' choices.

Making SuiteSignOn Integrations Available to Users

NetSuite administrators can enable SuiteSignOn integrations in their account by completing the following tasks. If you are not familiar with the SuiteSignOn feature, see Outbound Single Sign-on (SuiteSignOn) and Understanding SuiteSignOn.

**Warning:** NetSuite administrators should exercise caution when integrating with third-party applications using SuiteSignOn. Some integrations may require access to, or even modify, some data in your NetSuite account. Make sure you review the data requirements and understand what kind of information is accessed, retrieved, modified, or deleted by the third-party system. NetSuite has no control, responsibility, or liability regarding any third-party applications, even if NetSuite offers resale and integration options for customers' convenience. You use and integrate with third-party applications at your sole risk.

**Summary of tasks:**

1. Enable SuiteSignOn-related features (see SuiteSignOn Required Features).
2. Install a SuiteSignOn bundle (see Installing a SuiteSignOn Bundle).
3. Complete the implementation tasks required for making the third-party application available to NetSuite users (see Completing Account Setup for SuiteSignOn).

**Note:** The tasks mentioned here are aimed at account administrators. If you are an application provider and want to create a SuiteSignOn integration, see Setting Up SuiteSignOn Integration.

---

### Installing a SuiteSignOn Bundle

The main requirement to implement SuiteSignOn integration in your account is to install a bundle. The application provider that created the bundle should let you know the bundle name and ID. Also, they should indicate where you can find the bundle, either in the Bundle Repository or in an account. If you need to install a bundle directly from an account, you will need the account ID.

**To install a SuiteSignOn bundle:**

1. Go to Customization > SuiteBundler > Search & Install Bundles, and follow the instructions in Installing a Bundle.
2. Follow the instructions in the bundle documentation file to completely implement SuiteSignOn in your account. See Completing Account Setup for SuiteSignOn.

### Completing Account Setup for SuiteSignOn

After you have finished Installing a SuiteSignOn Bundle in your account, you must complete a few additional setup tasks in NetSuite.

The bundle documentation file should include instructions for these tasks. If you have not yet reviewed this file, go to Customization > SuiteBundler > Search & Install Bundles > List, and on the Installed Bundles page, click the Documentation link.

You should follow the instructions in this file. The list below describes tasks that are likely to be included in this file:

1. Go to Customization > SuiteBundler > SuiteSignOn, click the link for the newly installed SuiteSignOn integration, and verify that the SuiteSignOn page looks correct. The bundle documentation should include a checklist or a screenshot for you to use for this purpose.
2. Make changes to the SuiteSignOn page as necessary. You must have the SuiteSignOn permission to edit SuiteSignOn records.
   - If your account ID for the application provider is required for identification, enter it in the Partner Account field. This value is not necessary if the integrated application does not use it for identification. The bundle documentation should indicate whether this value is required.
   - If you want to control the level of NetSuite access for SOAP web services callbacks from integrated applications, change the Web Services Access option. The following options are available:
     - Same as UI Role - the default, which allows SOAP web services callbacks from integrated applications with the same level of permissions as in the user interface integration.
     - No Access - prevents integrated applications from accessing NetSuite through SOAP web services callbacks.
     - Additional options for any SOAP web services only roles in the account - selecting one of these roles allows SOAP web services callbacks from integrated applications, but limits access to the permissions levels assigned to the selected role.
As a security best practice, you should provide the minimum level of access required for SuiteSignOn integrated applications. For example, if an application only requires user interface integration, it is best to set the Web Services Access option to No Access.

This field is also available for viewing and editing on the SuiteSignOn list page at Setup > Integration > SuiteSignOn.

**Important:** Be aware that changing the Web Services Access option could possibly break an integration, because some integrations may depend on existing user permissions.

- If the bundle includes custom fields to be used as user identification, ensure that they appear on the User Identification subtab and are checked. The bundle documentation should indicate whether these fields are included.

Be aware that the names of bundled custom fields may be changed slightly from those listed in the bundle documentation, if any of their IDs conflict with preexisting custom fields in your account. If a conflict is detected, the bundled custom field ID is appended with ",#". For example, if a bundle installs a custom field with an ID of custentitybanana and a preexisting custom field has the same ID, the bundled field ID is changed to custentitybanana,2. This field ID also is changed where it is referenced in SuiteSignOn setup information, either in the Integration Variables field, or on the User Identification subtab.

**Warning:** NetSuite administrators should exercise caution when integrating with third-party applications using SuiteSignOn. Some integrations may require access to, or even modify, some data in your NetSuite account. Make sure you review the data requirements and understand what kind of information is accessed, retrieved, modified, or deleted by the third-party system. NetSuite has no control, responsibility, or liability regarding any third-party applications, even if NetSuite offers resale and integration options for customers’ convenience. You use and integrate with third-party applications at your sole risk.

3. You may need to populate values for custom fields used for user identification, through manual entry, mass update, CSV import, or another import process. Follow the bundle documentation instructions for this task.

4. If any portlet connection points are included, you must:
   - add one or more custom portlets that display the specified scripts to your dashboard and publish it to other users,
   - or:
   - provide instructions to users for adding custom portlets to their own dashboards.

   See Adding Custom Portlets for SuiteSignOn.

5. If you do not want a subtab connection point to be available to all users with access to the specified record type, you can create a custom form that hides the subtab, define this custom form as preferred for some users, and restrict their access to other forms for that record type. See the help topics Creating Custom Entry and Transaction Forms and Defining Preferred Entry and Transaction Forms.

### Adding Custom Portlets for SuiteSignOn

After you have installed a SuiteSignOn bundle, any subtab, Suitelet, and user event connection points are immediately available to users in your account. However, a portlet connection point is not available to users until they have added a custom portlet to the dashboard and configured that portlet to use the script defined for that connection point.
You can do either of the following to expose a portlet connection point:

- Add a custom portlet to your own dashboard and publish it to users. This option provides you with greater control.
  
  See the help topic Publishing Dashboards Overview. Be aware that you can only publish a dashboard to users with the same center as you, so you may need to log in with multiple roles and repeatedly add the custom portlet to multiple dashboards to make the portlet available to users with different centers.

- Provide users with instructions for adding a custom portlet to their dashboards.

**To expose a portlet connection point on your dashboard:**

1. On the page where you want to add the connection point, click the Customize this Page link.
2. In the Add Content panel, drag and drop a Custom Portlet object to the desired location on the page.
3. In the Custom Content portlet, click the Set Up link.
4. In the Set Up Scripted Content dialog, select the name of the script that is listed as the portlet connection point, and click Save.

**SuiteSignOn Definitions, Parameters, and Code Samples**

The SuiteSignOn feature uses a portion of the OAuth protocol specification. OAuth enables applications to access another application's protected resources from a web service through an API, without requiring users to disclose to the first application their credentials for the second application. The OAuth specification is available at http://oauth.net/documentation/spec.

See the following sections for more information:

- NetSuite SuiteSignOn Translation of OAuth Definitions
- Mappings of dc and env URL Parameter Values
- Sample SuiteSignOn HTTP Calls

**NetSuite SuiteSignOn Translation of OAuth Definitions**

Familiarize yourself with the OAuth 1.0 Protocol, RFC 5849. Refer to the following table to understand how the SuiteSignOn feature implements OAuth:

<table>
<thead>
<tr>
<th>NetSuite Term</th>
<th>Definition</th>
<th>Analogous OAuth Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Provider</td>
<td>NetSuite</td>
<td>Service Provider</td>
</tr>
<tr>
<td>Consumer</td>
<td>External application provider of the application to be accessed from NetSuite through SuiteSignOn, may also be known as partner.</td>
<td>Consumer</td>
</tr>
<tr>
<td>Consumer Key</td>
<td>Globally unique identifier of the consumer, generated by NetSuite.</td>
<td>Consumer Key</td>
</tr>
<tr>
<td>Shared Secret</td>
<td>Password used to establish ownership of the Consumer Key, entered by the Consumer when setting up the SuiteSignOn connection. The shared secret has a 1-1 relationship with the Consumer Key.</td>
<td>Consumer Secret</td>
</tr>
<tr>
<td>NetSuite Term</td>
<td>Definition</td>
<td>Analogous OAuth Term</td>
</tr>
<tr>
<td>--------------</td>
<td>------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Token</td>
<td>Value used to gain access to protected resources on behalf of the user, generated by NetSuite, good for a single session.</td>
<td>User</td>
</tr>
<tr>
<td>User</td>
<td>Individual who has logged into NetSuite and initiated activity between the consumer and NetSuite.</td>
<td>User</td>
</tr>
</tbody>
</table>

Mappings of dc and env URL Parameter Values

SuiteSignOn integration code dynamically creates values for the dc and env URL parameters.

The dc parameter ensures that integrations work for NetSuite accounts on all data centers.

The env parameter is used to identify whether the request is coming from a NetSuite production, release preview, or sandbox account. You might use this data to deny certain sessions (for example, requests coming from a sandbox account). Or you may want to map certain NetSuite requests to specific external applications. For example, you may want to map a request originating from the NetSuite sandbox account to a sandbox version of the external application.

The dc and env parameters apply the following:

- HTTP verify calls from external applications to NetSuite — see External Application HTTP Verify Call
- HTTP verify call responses from NetSuite to external applications — see NetSuite HTTP Verify Call Response

Both the dc and the env variables are required for your external application to map to a NetSuite domain.

For more information on dc and env parameter values, and URL examples, see the help topic Understanding NetSuite URLs.

Sample SuiteSignOn HTTP Calls

As described in SuiteSignOn Sequence Diagram and Connection Details, the SuiteSignOn handshake process between NetSuite and the external application includes the following calls in HTTP headers:

1. **NetSuite HTTP Outbound Call** sends the token and any context information to the external application.
2. **External Application HTTP Verify Call** returns the token and sends other required parameters to NetSuite.
   - This call requires an Authorization header in the OAuth 1.0 format.
3. **NetSuite HTTP Verify Call Response** sends user identity information in XML format to the external application.

NetSuite HTTP Outbound Call

When a user accesses a SuiteSignOn connection point, NetSuite issues an outbound call to start the handshake. The following is an example of this call:

```
GET /SSO/demoApp.php?
oauth_token=05016d16126a7a6c554656421e242310060807051b17ee54e6d26986d8aa&dc=001&env=PRODUCTION HTTP/1.1
Host: externalsystem.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:19.0) Gecko/20100101 Firefox/19.0
```
Be aware of the following:

- This call uses the GET method to send a generated token to the external application.
- The external application, not the local host, is the host.
- This call includes a dc and an env parameter. These values can be mapped to the correct domain for NetSuite access. SuiteSignOn integration code should dynamically populate domains for NetSuite access based on the value of the dc and env parameters, to ensure that integrations work for NetSuite accounts on all data centers, and all types of accounts (for example, production accounts as opposed to sandbox accounts). See Mappings of dc and env URL Parameter Values and Understanding NetSuite URLs for more information on dc and env parameters, and examples of URLs.
- This call also may include context information, if integration variables have been defined for the connection point on the NetSuite SuiteSignOn page. `customer_id=970` is an example of an integration variable. It could be included as a URL parameter, as follows:

  ```
  GET /SSO/demoApp.php?oauth_token=05016d16126a7a6c554656421e242310008007051b17ee54e6d26986d8aa&customer_id=970&dc=001&env=PRODUCTION HTTP/1.1
  ```

- URL parameters are separated by the ampersand (&). You must ensure that your code properly parses these parameters. Your code should not rely on the number or order of URL parameters, as these are subject to change.

### External Application HTTP Verify Call

Upon receipt of the NetSuite HTTP outbound call, the external application must issue an HTTP verify call. The following is an example of this call.

**Note:** You should use HMAC-SHA256, as it is the most secure signature option. You can also use HMAC-SHA1. PLAINTEXT is supported.

```plaintext
GET /app/common/integration/ssoapplistener.nl HTTP/1.0
Host: system.netsuite.com
Authorization: OAuth oauth_consumer_key="6OtBtQV4nmEOQKpw",
               oauth_token="05016d16126a7a6c554656421e242310008007051b17ee54e6d26986d8aa",
               oauth_nonce="kPeHzQpN6bZXsWu5w2nm",
               oauth_timestamp="1490706743",
               oauth_signature_method="HMAC-SHA256",
               oauth_version="1.0",
               oauth_signature="vh3C69af9EwXXGbmlDqEnA4ixYbtaM3kq9WH60it4e5Qh3D"
```

Be aware of the following, as shown in the example of the HTTP verify call:

- This call should use the GET method.
- This call should point to the NetSuite `ssoapplistener.nl` URL.
- The host domain should be dynamically populated with the system domain that maps to the dc and env parameter values. See Mappings of dc and env URL Parameter Values and Understanding NetSuite URLs for more information on dc and env parameters, and URL examples.
- This call should include the Authorization header. The entire Authorization header, including all of the parameters, must be in a single line. See The OAuth Authorization Header for Outbound SSO for more information.
The OAuth Authorization Header for Outbound SSO

The outbound HTTP Verify call should include the following parameters in the Authorization header. The entire header, including all of the parameters, must be in a single line. The CRLF character indicates the end of the header.

### Note:
For a description of the OAuth 1.0 protocol and signature validation, see the OAuth 1.0 Protocol, RFC 5849.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>oauth_token</td>
<td>The token generated and sent by NetSuite.</td>
<td></td>
</tr>
<tr>
<td>oauth_consumer_key</td>
<td>A globally unique identifier for the application provider, generated by NetSuite when the integration is set up on the SuiteSignOn page.</td>
<td></td>
</tr>
</tbody>
</table>
| oauth_signature_method | HMAC-SHA256 and HMAC-SHA1 are supported signature methods for ssoapplistener calls. | ▪ You should use HMAC-SHA256, as it is the most secure signature option.  
▪ You can also use HMAC-SHA1.  
▪ PLAINTEXT is supported. |
| oauth_signature     | The signature is computed based on chosen signature method.                 | The token secret mentioned in the OAuth 1.0 specification is an empty string, so the hashing key is:  
(shared_secret + & + ""
The shared secret should be percent-encoded.  
For more information about percent-encoding, go to https://tools.ietf.org/html/rfc5849#section-3.6.  
For more information about the shared secret, see:  
▪ the description of Shared Secret in Setting SuiteSignOn Basic Definitions.  
▪ Notes about Modifying the Shared Secret in Editing SuiteSignOn Records. |
| oauth_timestamp     | The number of seconds since January 1, 1970 00:00:00 GMT. The timestamp value must be a positive integer and must be equal to or greater than the timestamp used in previous verify calls. |       |
| oauth_nonce         | A random number that is unique across verify calls with the same timestamp value. |       |

### Generate the Signature for the OAuth Header for Outbound SSO

Some users have difficulty understanding how to construct a signature for the Authorization header. This is the header used in the External Application HTTP Verify Call.

For more information about generating the signature, see Troubleshooting SuiteSignOn (Outbound SSO)

The following input parameters for this example:

```bash
$url = "https://system.netsuite.com/app/common/integration/ssoapplistener.nl"
```
$oauth_consumer_key="6Ot8tQv4nmEOQKpw"
$oauth_consumer_secret = "P@ssw0rd 123"; //shared secret
$oauth_token="030f6c1db6b106c6b445655477e72571343582efefc809d"
$oauth_nonce="kPeHzQn6BZxsm5uW2rm"
$oauth_timestamp="1490706743"
$oauth_signature_method="HMAC-SHA256"
$oauth_version="1.0"

This example uses the PHP OAuth library. For more information, see https://tools.ietf.org/html/rfc5849#section-3.4.1.

To generate the oauth_signature:

1. Construct a base string for the signature.

   $baseString = oauth_get_sbs($httpMethod, $url, array('oauth_consumer_key' => $oauth_consumer_key,
   'oauth_nonce' => $oauth_nonce,
   'oauth_signature_method' => $oauth_signature_method,
   'oauth_timestamp' => $oauth_timestamp,
   'oauth_token' => $oauth_token,
   'oauth_version' => $oauth_version));

   For more information, see Create the Base String Manually in Troubleshooting SuiteSignOn (Outbound SSO).

2. The signature key is used to sign the base string in the HMAC-SHA algorithm. The key is constructed from the URL-encoded value for the consumer secret, with the ampersand character (&) as the delimiter.

   $key = rawurlencode($oauth_consumer_secret) . "&" . "";

3. The signature is a base64 value of the HMAC-SHA, where the message is Base String and key is the key from the previous step.

   $signature = base64_encode(hash_hmac('sha256', $baseString, $key, true)); //or sha1 or plaintext
   // signature for this example: 1/3WKQsNRU4/EupyUWMciPRmEHaQEYCL7afJCLmMnd4=

Authorization: OAuth oauth_token="030f6c1db6b106c6b445655477e72571343582efefc809d",
oauth_consumer_key="6Ot8tQv4nmEOQKpw", oauth_nonce="kPeHzQn6BZxsm5uW2rm",
oauth_timestamp="1490706743", oauth_signature_method="HMAC-SHA256", oauth_version="1.0",
oauth_signature="1%2F3WKQsNRU4%2FEupyUWMciPRmEHaQEYCL7afJCLmMnd4%3D"

NetSuite HTTP Verify Call Response

Upon receipt of the verify call from the external application, NetSuite sends a response. The following is an example of this response:

HTTP/1.1 200 OK
Date: Tue, 16 Apr 2016 13:30:41 GMT
Server: Apache/2.2.17
Set-Cookie: lastUser=1326288_79_3; expires=Tuesday, 23-Apr-2016 13:30:42 GMT; path=/
Set-Cookie: NS_VER=2015.2.0; domain=system.netsuite.com; path=/
X-Powered-By: Servlet/2.5 JSP/2.1
P3P: CP="CAO PSAa OUR BUS PUR"
Troubleshooting SuiteSignOn (Outbound SSO)

This section includes the following troubleshooting information:

- SuiteSignOn (Outbound SSO) Error Messages
- Troubleshooting the SuiteSignOn Signature
- Creating the Authorization Header for SuiteSignOn
- The Base String for SuiteSignOn

SuiteSignOn (Outbound SSO) Error Messages

When SuiteSignOn (Outbound SSO) authentication fails, it returns a **WWW-Authenticate** header with the details of the failure. Look for the parameter **oauth_problem**.

HTTP response header with error Example

```
```

The error codes and meanings are defined in the following table.

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Problem</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>consumer_key_rejected</td>
<td>No SuiteSignOn application with this key was found.</td>
<td>Ensure the consumer key is correct. If there are no SuiteSignOn applications set up, create a new one.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Problem</td>
<td>Resolution</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>parameter_absent</td>
<td>The Authorization header does not contain all necessary parameters.</td>
<td>Examine the <code>oauth_parameters_absent</code> parameter for more information on which parameter is missing.</td>
</tr>
<tr>
<td>parameter_rejected</td>
<td>The same parameter was sent multiple times.</td>
<td>Examine the <code>oauth_parameters_rejected</code> parameter for more information on which parameter was rejected.</td>
</tr>
<tr>
<td>signature_invalid</td>
<td>The request was not signed correctly.</td>
<td>See <a href="#">Generate a Signature</a> for the correct method of signing a request.</td>
</tr>
<tr>
<td>signature_method_rejected</td>
<td>The algorithm used to create signature is not supported.</td>
<td>The only supported algorithms are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- You should use HMAC-SHA256, as it is the most secure signature option.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- You can use HMAC-SHA1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- PLAINTEXT is supported.</td>
</tr>
<tr>
<td>timestamp_refused</td>
<td>The timestamp of the request must be within plus or minus five (+ or –5) minutes of the server time.</td>
<td>Ensure that:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Your computer clocks are synchronized using the NTP protocol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Requests are sent soon after generating the authorization header.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Requests are not being queued before being sent to NetSuite.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to the parameter <code>oauth_acceptable_timestamps</code> for the accepted range of the timestamp.</td>
</tr>
<tr>
<td>token_expired</td>
<td>The token could not be found.</td>
<td>Ensure that:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The token is correct.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The user is still logged in the NetSuite UI in the same role. The token is only valid until the user changes roles or logs out of the UI.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- The user still has access to the NetSuite UI.</td>
</tr>
<tr>
<td>version_rejected</td>
<td>The <code>oauth_version</code> is unknown.</td>
<td>The only accepted value for <code>oauth_version</code> is 1.0.</td>
</tr>
</tbody>
</table>

**Troubleshooting the SuiteSignOn Signature**

This section covers generating a valid signature.

**Note:** The values defined in this section are the values used in the examples in the following sections.

**Generate a Signature**

Some users have difficulty constructing a valid signature. There are many ways to generate a signature for SuiteSignOn (Outbound SSO). This is one example of how to do it correctly.
The following sections describe how to correctly create a signature. There are PHP examples for each step.

- **Input Parameters for the Example**
- **Step 1: Construct a Base String for the Signature**
- **Step 2: Signature Key**
- **Step 3: Signature**

**Note:** All encoding in SuiteSignOn (Outbound SSO) is percent-encoding. For more information about percent-encoding, go to [https://tools.ietf.org/html/rfc5849#section-3.6](https://tools.ietf.org/html/rfc5849#section-3.6). The examples in this section use PHP rawurlencode.

### Input Parameters for the Example

These are the input parameters used for this example.

```php
$url = 'https://system.netsuite.com/app/common/integration/ssoapplclistener.nl';
$httpMethod = 'GET';
$tokenKey = '030e6a121766126c6b4456555477e7252517c395926f3430a';
$tokenSecret = ''; // Outbound SSO does not use token secret
$consumerKey = 'VutaTaro1kt0XKoD';
$consumerSecret = 'S3cr3t P@ssw0rd'; // In UI called "Shared secret"
$signatureMethod = 'HMAC-SHA256'; // or HMAC-SHA1 or PLAINTEXT
$nonce = 'fjaLirs1cC GV 2k8x0pg'; //
$timestamp = '1508242306'; // time();
$version = '1.0';
```

### Step 1: Construct a Base String for the Signature

The first step in creating signature is constructing a Base String.

**Note:** This step is not needed when using PLAINTEXT as a signature method.

**Base String Creation**

```php
$baseString = oauth_get_sbs($httpMethod, $url, array('oauth_consumer_key' => $consumerKey,  
'oauth_nonce' => $nonce,  
'oauth_signature_method' => $signatureMethod,  
'oauth_timestamp' => $timestamp,  
'oauth_token' => $tokenKey,  
'oauth_version' => $version));
```

**Base String Example**

```
GEThttps%3A%2F%2Fsystem.netsuite.com%2Fapp%2Fcommon%2Fintegration%2Fssoapplclistener.nl%3Doauth_consumer_key%3DVutaTaro1kt0XKoD%26oauth_nonce%3DfjaLirs1cC GV 2k8x0pg%26oauth_signature_method%3DHMAC-SHA256%26oauth_timestamp%3D1508242306%26oauth_token%3D030e6a121766126c6b4456555477e7252517c395926f3430a%26oauth_version%3D1.0
```
Troubleshooting SuiteSignOn (Outbound SSO)

Note: The examples use the oauth library. The command for installing the library is `sudo pecl install oauth`. See https://tools.ietf.org/html/rfc5849#section-3.4.1 for more information on the signature base string.
See also Create the Base String Manually.

Step 2: Signature Key

**Important:** The signature key must be percent-encoded as specified in https://tools.ietf.org/html/rfc5849#section-3.4.1.

The signature key is used to sign the base string in the HMAC-SHA algorithm. The key is constructed from the URL-encoded values for:
- consumer secret and
- token secret (empty string)
- with the ampersand character (&) as the delimiter

```php
$key = rawurlencode($consumerSecret) . '&' . rawurlencode($tokenSecret);
```

Step 3: Signature

HMAC-SHA

**Signature HMAC-SHA Example**

```php
$signature = base64_encode(hash_hmac('sha256', $baseString, $key, true));
// $signature = base64_encode(hash_hmac('sha1', $baseString, $key, true));
```

The signature is a base64 value of the HMAC-SHA, where the message is Base String and key is the key from the previous step.

**Signature HMAC-SHA256 Example**

`PP1VMUdgDJeSkeNwJ8EqjKow0Vdd5ey9JqgRT3MwQJi4c=`

**Signature HMAC-SHA1 Example**

`6nMUbMr8c8sfVDoOYmsB6wmpo=`

PLAINTEXT

**Signature PLAINTEXT**

```php
$signature = $key;
```

**Signature PLAINTEXT Example**

`S3cr3t%20P%40ssw0rd&`
Creating the Authorization Header for SuiteSignOn

The creation of the header is straightforward. Put the correct parameter in the correct place.

**Important:** Each parameter must be percent-encoded. The examples in this section use PHP rawurlencode.

**Header**

```php
$header = 'Authorization: OAuth ' .
    'oauth_token="'.rawurlencode($tokenKey).'"', 'oauth_consumer_key="'.rawurlencode($consumerKey).'"', 'oauth_nonce="'.rawurlencode($nonce).'"', 'oauth_timestamp="'.rawurlencode($timestamp).'"', 'oauth_signature_method="'.rawurlencode($signatureMethod).'"', 'oauth_version="'.rawurlencode($version).'"', 'oauth_signature="'.rawurlencode($signature).'"';
```

**Header HMAC-SHA256 Example**

```
Authorization: OAuth oauth_token="030e6a121766126c6b445655477e7252517c395926f3430a", oauth_consumer_key="VutaTaro1kltGNOXO", oauth_nonce="fjalrisIcCGVz8X0qg", oauth_timestamp="1508242306", oauth_signature_method="HMAC-SHA256", oauth_version="1.0", oauth_signature="Q6jMu61V%2BORdf6UeZ39ixFSu3rX02dwaucQg8P1cWq5Q3D"
```

**Header HMAC-SHA1 Example**

```
Authorization: OAuth oauth_token="030e6a121766126c6b445655477e7252517c395926f3430a", oauth_consumer_key="VutaTaro1kltGNOXO", oauth_nonce="fjalrisIcCGVz8X0qg", oauth_timestamp="1508242306", oauth_signature_method="HMAC-SHA1", oauth_version="1.0", oauth_signature="AAt58FZt8gxQZz9gtxSF%2FErFbcg%3D"
```

**Header PLAINTEXT Example**

```
Authorization: OAuth oauth_consumer_key="VutaTaro1kltGNOXO", oauth_token="030e6a121766126c6b445655477e7252517c395926f3430a", oauth_nonce="fjalrisIcCGVz8X0qg", oauth_timestamp="1508242306", oauth_signature_method="PLAINTEXT", oauth_version="1.0", oauth_signature="S3cr3t%2520P%2540ssw0rd%26"
```

**Additional Shared Secret Requirements If Using PLAINTEXT**

The shared secret must comply with the requirements specified in RFC 5849- OAuth 1.0, sections 3.4.4, 3.5.1 and 3.6.

- **The Shared Secret must be percent-encoded.** Percent-encoding uses hexadecimal numbers. (You may be more familiar with URL encoding, which is different than percent-encoding. In percent-encoding, the space character (+) must be encoded as `%20`. When double-encoded, the space character `%20` becomes `%2520`.)
- The OAuth signature must include the ampersand character (`&`) which is used as a delimiter (ASCII code 38 in decimal, but `%26` after encoding) even if the token secret is not used in SuiteSignOn.
- For SuiteSignOn, the format is: `signature = rawurlencode( rawurlencode(shared secret) ' & ' )`
For example, if you chose P@mpered15! as your shared secret, when encoded, the signature would be: "P%2540mpered15%2521%26"

The Base String for SuiteSignOn

The first step in creating a signature is construction of the Base String.

**Note:** Constructing a Base String is not necessary if you are using PLAINTEXT as the signature method. However, rather than PLAINTEXT, you should use HMAC-SHA256, as it is the most secure signature option or you can use or HMAC-SHA1.

The values used in the following code samples are defined in the section Troubleshooting the SuiteSignOn Signature.

See the following topics in this section:

- Create the Base String Manually
- The restletBaseString Function

Create the Base String Manually

In the following example, the Base String consists of three parts. Each step contains an image of a piece of the code to show the line numbers. To view the entire code example (without line numbers) see the following section: The restletBaseString Function.

**Note:** POST parameters are used only with content type "application/x-www-form-urlencoded". However, this content type is not allowed by RESTlets.

1. HTTP method - line 3

   **Note:** The HTTP method must be in uppercase.

2. URL - lines 6-16
   - URL is taken without parameters. (lines 6-12)
   - Schema (http, https) and hostname must be in lowercase. (lines 13-15)

3. Parameters - lines 19-51
Put all OAuth, GET, and POST parameters into the array of arrays. (lines 19-37)
Parameter names and values are urldecoded before entering into array (lines 30–34)
The array is alphabetically sorted by parameter name. (line 40)
The string containing all parameters is created. Each name and value is separated by the equal character (=) and each pair is separated by the ampersand character (&). Both name and value are rawurlencoded. (lines 42-50)
The whole string containing parameters is rawurlencoded before joining with rest of the Base String (line 51)

```php
$params = array();
$params['oauth_consumer_key'] = array($consumerKey);
$params['oauth_token'] = array($tokenKey);
$params['oauth_nonce'] = array($nonce);
$params['oauth_timestamp'] = array($timestamp);
$params['oauth_signature_method'] = array($signatureMethod);
$params['oauth_version'] = array($version);

foreach (explode('&', $getParams . '&'. $postParams) as $param) {
    $parsed = explode('=', $param);
    if ($parsed[0] === '') {
        $value = isset($parsed[1]) ? urldecode($parsed[1]) : '';
        if (isset($params[urldecode($parsed[0])])) {
            array_push($params[urldecode($parsed[0])], $value);
        } else {
            $params[urldecode($parsed[0])] = array($value);
        }
    }
}

// all parameters must be alphabetically sorted
ksort($params);

$paramsString = '';
foreach ($params as $key => $valueArray) {
    // all values must be alphabetically sorted
    sort($valueArray);
    foreach ($valueArray as $value) {
        $paramsString .= rawurlencode($key) . '=' . rawurlencode($value) . '&';
    }
}
$paramsString = substr($paramsString, 0, -1);
$baseString = rawurlencode($paramsString);
return $baseString;
```

### The restletBaseString Function

```php
def function restletBaseString($httpMethod, $url, $consumerKey, $tokenKey, $nonce, $timestamp, $version, $signatureMethod, $postParams){
    // http method must be upper case
    $baseString = strtolower($httpMethod) . '&';

    // include url without parameters, schema and hostname must be lower case
    if (strpos($url, '?')){
        $baseUrl = substr($url, 0, strpos($url, '?'));
        $getParams = substr($url, strpos($url, '?') + 1);
    } else {
        $baseUrl = $url;
```
$getParams = "";
}
$hostname = strtolower(substr($baseUrl, 0, strpos($baseUrl, '/', 10)));
$path = substr($baseUrl, strpos($baseUrl, '/', 10));
(baseUrl = $hostname . $path;
$baseString .= rawurlencode($baseUrl) . ' ';

// all oauth and get params. First they are decoded, next alphabetically sorted, next each key and values is encoded and finally whole parameters are encoded
$params = array();
$params['oauth_consumer_key'] = array($consumerKey);
$params['oauth_token'] = array($tokenKey);
$params['oauth_nonce'] = array($nonce);
$params['oauth_timestamp'] = array($timestamp);
$params['oauth_signature_method'] = array($signatureMethod);
$params['oauth_version'] = array($version);

foreach (explode('&', $getParams . ". $postParams) as $param) {
    $parsed = explode('=', $param);
    if ($parsed[0] != "") {
        $value = isset($parsed[1]) ? urldecode($parsed[1]): "";
        if (isset($params[urldecode($parsed[0])])) {
            array_push($params[urldecode($parsed[0])], $value);
        } else {
            $params[urldecode($parsed[0])] = array($value);
        }
    }
}

// all parameters must be alphabetically sorted
ksort($params);

$paramString = "";
foreach ($params as $key => $valueArray){
    // all values must be alphabetically sorted
    sort($valueArray);
    foreach ($valueArray as $value){
        $paramString .= rawurlencode($key) . '='. rawurlencode($value) . ' ';
    }
}
$paramString = substr($paramString, 0, -1);
(baseUrl = rawurlencode($paramString);
return $baseString;
Inbound Single Sign-on

The NetSuite inbound single sign-on feature allows users to go directly from an external user-authenticating application to NetSuite, without having to log in separately to NetSuite. This feature allows a one-way trust relationship to be established between the external application and NetSuite, so that after users present login credentials to the external application, they can gain access to NetSuite as well.

Inbound single sign-on access generally has two types of users:

- Customers, who want to integrate their own NetSuite data with an external application’s data.
- Application providers, who want to integrate customer data stored in their data center with their customers’ NetSuite data.

With inbound single sign-on, authentication information from the external application is passed to NetSuite through an encrypted token, and a dynamically constructed URL redirects users from the external site to a NetSuite landing page. A mapping between each user’s external credentials and their NetSuite credentials is created, either through a SOAP web services operation or through the user’s login to NetSuite on their first single sign-on access.

To implement inbound single sign-on from your site to NetSuite, you must set up a trust relationship, with OpenSSL encryption keys, between your application and NetSuite. These keys are used to produce and interpret encrypted tokens. You also must write application code that dynamically constructs the redirect URL for each inbound single sign-on user. HTTP POST requests are not supported. NetSuite provides a downloadable kit with tools you can use for these tasks.

To get started with inbound single sign-on:

**Important:** As of 2018.1, new solutions using Inbound SSO for SOAP web services are not supported. Use Token-based Authentication (TBA) instead when creating new inbound SSO solutions for SOAP web services. See Token-based Authentication and Web Services.

1. Review this guide, including the following, to ensure that the NetSuite inbound single sign-on feature will meet your needs.
   - For an overview of how inbound single sign-on works, see the help topic Understanding Inbound Single Sign-on.
   - For instructions for setting up and implementing an inbound single sign-on integration, see the help topic Setting Up Inbound Single Sign-on. This section includes the following information:
     - Initial Setup for the Inbound Single Sign-on Feature
     - Implementing Inbound Single Sign-on in an External Application
     - Generating Keys Using OpenSSL
     - Creating the Initial Mapping of the Administrator Role for Inbound Single Sign-on
     - Creating Single Sign-on Code Using SSOUId
   - For instructions for setting up inbound signle sign-on mappings, see the help topic Mapping Users and Roles for Inbound Single Sign-on Access to NetSuite
   - For technical background, see the help topic Technical Summary of Inbound Single Sign-on.

2. Be aware of the following:
   - Inbound single sign-on access is supported for the NetSuite application, including the Customer Center, and for NetSuite web stores.
   - Using the Administrator role to log in to a web store is not supported.
If a user must access both Customer Center and non-Customer Center roles, they must have at least two mappings. Separate mappings are required for each Customer Center role.

- Inbound single sign-on access to the Customer Center is supported for NetSuite users classified as customers and for customer contacts.
- If a user must access both Customer Center and non-Customer Center roles, they must have at least two mappings. Separate mappings are required for each Customer Center role.
- Inbound single sign-on access to NetSuite respects IP address restriction rules. For information about this feature, see Enabling and Creating IP Address Rules.
- Inbound single sign-on access to web store is supported for custom checkout domains, multi-site implementations, and sites customized with SSP applications. Access is also supported for Reference Cart & One Page Checkout, the NetSuite reference implementation of the web store checkout process. See the help topic Inbound Single Sign-on Access to Web Store.

3. After you have confirmed that you want to implement inbound single sign-on, contact your account manager to purchase the feature.

Alternate Inbound Single Sign-on Mechanisms

**Important:** As of 2018.1, new solutions using Inbound SSO for SOAP web services are not supported. Use Token-based Authentication (TBA) instead when creating new inbound SSO solutions for SOAP web services. See Token-based Authentication and Web Services.

NetSuite supports other features that do not use this NetSuite version of inbound single sign-on authentication:


**Note:** It is not necessary to purchase the NetSuite Inbound Single Sign-on feature if you want to implement SAML Single Sign-on in NetSuite.

- The OpenID Single Sign-on feature supports inbound single sign-on access from Google Apps to NetSuite, relying on Google Accounts as the trusted system of authentication. See OpenID Single Sign-on.

Inbound Single Sign-on Overview

The NetSuite inbound single sign-on feature enables users to move directly from an external user-authenticating web application to NetSuite without additional authentication. This feature provides token-based integration.
The external application uses an encrypted token to pass the user’s identity to NetSuite. NetSuite verifies the token, then logs in the user. This single sign-on mechanism can be implemented through a link in the external application user interface, or through SOAP web services calls that use the `ssoLogin` operation.

### Understanding Inbound Single Sign-on

The following steps outline how inbound single sign-on to NetSuite works.

1. In most cases, a user initiates inbound single sign-on access to NetSuite by clicking a link in an authenticated area of an external site. This site can be used in conjunction with either the NetSuite application user interface or a NetSuite web store.

   ![Diagram](image)

   **Note:** Using the Administrator role to log in to a web store is not supported.

2. When a user initiates inbound single sign-on access, the external application produces a token that includes the following information:

   ![Diagram](image)

   **Important:** As of 2018.1, new solutions using Inbound SSO for SOAP web services are not supported. Use Token-based Authentication (TBA) instead when creating new inbound SSO solutions for SOAP web services. See [Token-based Authentication and Web Services](#).
the user’s external application company ID
This value is a string used by the external application to determine the company with which a user is associated, for example ABCAutoParts. It cannot contain spaces.

the user’s external application user ID
This value is a string used by the external application as a user identifier, for example John.Smith. It cannot contain spaces.

the current timestamp
The timestamp string is a decimal representation of the number of milliseconds since January 1, 1970, 00:00:00 GMT.

For more information about the token, see the following topics:
- Tables of Single Sign-on Redirect URL Parameters
- Elements of the Authentication Token String
- Example Inbound Single Sign-on Token

3. The external application encrypts the information included in the token.
   - To encrypt the token, the external application must have access to a private key generated using OpenSSL. To interpret the encrypted token, NetSuite must have access to a public key extracted from this private key.
   - The inbound single sign-on kit includes Java classes you can use to produce the private and public keys. For instructions, see the help topic Generating Keys Using OpenSSL.

4. After encryption of the token, the external application causes the user’s browser to perform a redirect to NetSuite. HTTP POST requests are not supported.
   - The redirect is either to the NetSuite application or to the web store, based on the target set in the code (either app or site).
   - The redirect uses a URL constructed specifically for this inbound single sign-on access. This URL includes required parameters such as:
     - a hex-encoded, encrypted string representing the token
     - the unique partner ID assigned by NetSuite Customer Support
     - For NetSuite application access only, the remote company ID, which is the company ID used in the token
     - For web store access only, the domain, NetSuite company ID, and site ID

   **Note:** For more information on required and optional parameters, see the help topic Tables of Single Sign-on Redirect URL Parameters.

   - This URL is valid for up to 15 minutes after the timestamp included in the encrypted token.
   - The inbound single sign-on kit includes a Java class you can use to create code that dynamically constructs this URL. For instructions, see the help topic Creating Single Sign-on Code Using SSOUrl.

5. NetSuite receives the token. Based on a unique partner ID assigned by NetSuite when inbound single sign-on is set up, NetSuite determines the public key that should be used to decrypt the token. After the token is decrypted, if the timestamp is valid, the request is honored.

6. NetSuite checks for a user mapping between the external application and NetSuite.
   - If there is an existing mapping, the user is logged in as defined by the mapping.
   - If a mapping does not exist, the user is prompted to provide NetSuite credentials to create the mapping.
Understanding Inbound Single Sign-on

Important: A user with an Administrator role must create the initial mapping from the external application to NetSuite. For more information, see the help topic Creating the Initial Mapping of the Administrator Role for Inbound Single Sign-on.

After the initial mapping to the Administrator role is completed:

- For web store access, the administrator is required to use the SOAP web services mapSso operation to create the account mapping for multiple users so that it is available before users initiate single sign-on access. However, using the Administrator role to log in to a web store is not supported.
- For NetSuite access, the administrator can use the SOAP web services mapSso operation to create the account mapping for multiple users, or can instruct users to create their own mappings. See the help topic Mapping Users and Roles for Inbound Single Sign-on Access to NetSuite.

7. After the user's identity has been verified, a NetSuite landing page displays.

   - The default landing page for NetSuite application access is the user's home page.
   - The default landing page for web store access is the site home page.
   - A non-default landing page can be defined by adding a landingurl parameter to the redirect URL.
   - If no mapped NetSuite identity is found, the NetSuite inbound single sign-on login page displays.

8. A NetSuite session initiated through inbound single sign-on is subject to standard NetSuite session timeout rules.

   - By default, the user is redirected to an inbound single sign-in login page on session timeout or error.
   - This NetSuite login page can be hidden by setting the redirect URL's hideloginpage parameter to true, so that the user is returned to a different page, such as an external application page. In this case, a returnurl parameter also must be added, to specify this alternate page.

9. The user can log out from an inbound single sign-on session in the same manner as any other NetSuite session.

   - By default, the user is redirected to the inbound single sign-in login page on logout.
   - If the redirect URL includes the returnurl parameter, the user is redirected to the page specified by this parameter instead. It is not necessary to set the hideloginpage parameter to T (true) to vary the page on logout.

Setting Up Inbound Single Sign-on

Important: As of 2018.1, new solutions using Inbound SSO for SOAP web services are not supported. Use Token-based Authentication (TBA) instead when creating new inbound SSO solutions for SOAP web services. See Token-based Authentication and Web Services.

See the following two procedures for important information:

1. Initial Setup for the Inbound Single Sign-on Feature

   This required procedure outlines requirements for setting up the inbound single sign-on feature in your account. It is guided by NetSuite Customer Support, and occurs after you have contacted your NetSuite account representative and purchased the Inbound Single Sign-on feature.
2. **Implementing Inbound Single Sign-on in an External Application**

   This procedure outlines options for inbound single sign-on integration from an external application to NetSuite.

   **Note:** It is not necessary to purchase the NetSuite Inbound Single Sign-on feature if you want to implement SAML Single Sign-on in NetSuite. For more information, see [SAML Single Sign-on](#). See also [Alternate Inbound Single Sign-on Mechanisms](#).

**Initial Setup for the Inbound Single Sign-on Feature**

**Warning:** You must contact your NetSuite account representative and purchase the inbound single sign-on feature to properly initiate the setup process. Do not attempt to complete these steps on your own. Wait until you are contacted by NetSuite Customer Support to begin the initial setup of this feature.

**To complete the initial setup of the inbound single sign-on feature in your account:**

1. Contact your account representative to purchase the inbound single sign-on feature.

   **Important:** As of 2018.1, new solutions using Inbound SSO for SOAP web services are not supported. Use Token-based Authentication (TBA) instead when creating new inbound SSO solutions for SOAP web services. See [Token-based Authentication and Web Services](#).

   a. NetSuite Customer Support will open a new support case, and contact you for specific information.
   b. NetSuite Customer Support will ask you to generate a public and private key pair using OpenSSL. See the help topic [Generating Keys Using OpenSSL](#).
   c. NetSuite Customer Support will ask you to provide the generated public key through the support case.
   d. NetSuite Customer Support will associate the public key with an Inbound Single Sign-on Partner ID, and will provide this unique Partner ID to you.

2. After NetSuite Customer Support guides you through the initial setup, and provides you with your unique Partner ID, you will be ready to implement inbound single sign-on in your application. See the help topic [Implementing Inbound Single Sign-on in an External Application](#).

**Implementing Inbound Single Sign-on in an External Application**

Before you attempt to implement any of the following options in your external application, you must have already contacted your NetSuite account administrator and purchased the Inbound Single Sign-on feature. Also, NetSuite Customer Support must have already guided you through the steps outlined in [Initial Setup for the Inbound Single Sign-on Feature](#).

**To implement inbound single sign-on in an external application:**

You can choose any of the following options to implement inbound single sign-on from an external application to NetSuite:

1. Download the kit for implementing inbound single sign-on:
   
Setting Up Inbound Single Sign-on

2. Add the ssov3.jar file from this kit to your Java classpath. You need the contents of this .jar file to facilitate compilation of your single sign-on integration code and to generate keys for token encryption.

**Note:** Java developers can add the ssov3.jar to your classpath, along with the Java run-time environment classes. Source code is also provided for developers in non-Java environments as a template for implementation.

3. Write application code that dynamically constructs redirect URLs to be used when users initiate inbound single sign-on access. HTTP POST requests are not supported. See the help topic Creating Single Sign-on Code Using SSUrl.

4. Write SOAP web services code for the single sign-on integration as needed. You can programmatically initiate access with ssoLogin, and/or programmatically map users' external credentials to NetSuite credentials. See the help topic SOAP Web Services Single Sign-on Operations.

5. Provide error handling for status codes returned from NetSuite inbound single sign-on sessions. See the help topic Error Handling for Inbound Single Sign-on.

6. To prevent single sign-on users from directly logging in to NetSuite, create a custom role that is designated as Single Sign-on Only, and assign this role to single sign-on users. See the help topic Setting Up a Single Sign-on Only Role.

Generating Keys Using OpenSSL

As described in the section Initial Setup for the Inbound Single Sign-on Feature, NetSuite Customer Support will ask you to generate a public and private key pair using OpenSSL. The public key is provided to NetSuite, for use in creating your unique Partner ID. You will use the private key in your implementation to encrypt authentication tokens.

**To generate keys for inbound single sign-on:**

1. Either append the openssl subdirectory provided in the inbound single sign-on kit to your PATH, or download source code from http://www.openssl.org/source.
   
   The binaries included in the openssl directory are derived from openssl0.9.6.tar.gz. If you are creating your own binaries from a downloaded source package, follow directions in the INSTALL file appropriate to your operating system.

2. After openssl is installed and in your PATH, type openssl to get the following prompt:

   OpenSSL>

3. At the prompt, use the following command to generate a private key:

   OpenSSL> genrsa -out <privKey.pem> -rand <f1><s><f2><s><f3><s><f4><s><f5> 2048

   - `<privKey.pem>` is the desired name of the output file.
   - `<f1>` through `<f5>` are names of files used as random seeds.
   - `<s>` is a separator:
     - `;` for Windows

Authentication Guide
Setting Up Inbound Single Sign-on

1. Convert the private key to DER using the Pem2Der class provided by NetSuite, by typing the following Java command:

```
java com.netledger.forpartners.encryption.Pem2Der <privKey.pem> <privKey.der>
```

2. Extract the public key from the private key using the Priv2Pub class provided by NetSuite, by typing the following command:

```
java com.netledger.forpartners.encryption.Priv2Pub <privKey.der> <pubKey.der>
```

Note: <privKey.der> and <pubKey.der> of this last command are your public and private keys.

Creating the Initial Mapping of the Administrator Role for Inbound Single Sign-on

The initial single sign-on account mapping must be a mapping to an Administrator role in NetSuite. This Administrator role mapping serves as authorization that NetSuite trusts the external authentication system. This requirement gives NetSuite administrators control over when single sign-on functionality is available to their users.

Note: The following procedure assumes that you have completed all tasks as described in the Initial Setup for the Inbound Single Sign-on Feature and Generating Keys Using OpenSSL topics. (You have added the ssov3.jar file from the inbound single sign-on kit to your Java classpath, NetSuite Customer Support has assisted with the generation of the public and private keys, and NetSuite Customer Support has provided you with the Partner ID for this account.)

A user with an Administrator role in this account must create the initial mapping between NetSuite and the external authentication system. To generate the token necessary to complete the initial mapping, you can use the tool included in the Inbound Single Sign-on kit to generate the URL. The URL includes the token necessary for the following procedure. The initial mapping process can be built in to your application to provide users with a smoother user experience (that is, by using an embedded browser to finish the mapping).

Note: You cannot use the SOAP web services mapSso operation to create the mapping for an Administrator role.

To create a token with the ssov3.jar file:

1. Call the ssov3.jar file.
2. Specify the appropriate parameters:
   a. \( rc \) = the target account number.
   b. \( p \) = the Partner ID assigned by NetSuite Customer Support.
   c. \( ru \) = the Administrator role to be used for the initial mapping.
   d. \( t \) = the type of URL you want to generate, either \( app \) (for inbound access to the NetSuite UI) or \( site \) (for inbound access to your website).

   See the help topic Tables of Single Sign-on Redirect URL Parameters for more information.

3. The tool generates a URL with a token in the form: 
   \(<domain>/app/login/secure/sso.nl<partnerID><TokenBigLongString>\)

   **Note:** The token is valid for 15 minutes. You must complete the following steps before the token expires, or generate a new token.

4. Copy and paste the URL into your browser's address bar and click Enter. The NetSuite Partner Login page displays.

5. On the NetSuite Partner Login page, enter your NetSuite email address and password.

6. Click Log in. The NetSuite Choose a role to create the mapping page displays.

7. Click your Administrator role for this account.

   **Note:** If you have Administrator roles in more than one account, ensure you are selecting the correct Administrator role for this specific account.

   The initial mapping of the Administrator role is now complete.

   After the initial mapping is completed, other users and roles can now be mapped to the external application. The SOAP web services mapSso operation can be used to create the account mappings for multiple users so that they are available before users initiate single sign-on access. (This method of mapping is required for web store access.)

   **Note:** Using the Administrator role to log in to a web store is not supported.

---

### Creating Single Sign-on Code Using SSOUrl

The quickest way to create inbound single sign-on code is to use the SSOUrl Java class provided in the downloadable kit. This class is available to you after you have downloaded the kit and added the ssov3.jar file to your Java classpath.

The SSOUrl.java file provides a template for Java code, along with explanatory comments. You can use this file to guide your creation of single sign-on integration code in Java. A command-line utility that you can run from a shell is also provided as an alternative.

**Note:** If you are NOT using the NetSuite SSOUrl class to generate redirect URLs, then you will need to construct them using your own methods from the base elements described in SSOUrl.java.

If you are using the SSOUrl class to implement inbound single sign-on integration, your application code needs to do the following:

- Initialize the SSOUrl class with the file name of the private key used to encrypt the authentication token.
Setting Up Inbound Single Sign-on

- Set the target of the inbound single sign-on access to either the NetSuite application (app) or the web store (site), so that the base URL for the integration is correctly generated:
  - for the NetSuite application:
    
    ```
    https://system.netsuite.com/app/loginsecure/sso.nl
    ```
    HTTP POST requests are not supported.
  - for the web store:
    
    ```
    https://checkout.mycompanystore.com/app/site/backend/sitesso.nl
    https://checkout.netsuite.com/app/site/backend/sitesso.nl
    https://checkout.na1.netsuite.com/app/site/backend/sitesso.nl
    https://checkout.na2.netsuite.com/app/site/backend/sitesso.nl
    ```

- For inbound single sign-on access to web store, set the domain for your web store.
  - If you have a custom checkout domain, set the domain appropriately.
  - If you are using a NetSuite-hosted checkout domain, set it to the appropriate checkout domain for your data center. The NetSuite company ID and site ID URL parameters are also required for web store access. For more information, see the help topic Web Store Access Only Parameters.

- Provide the single sign-on link as a link to an internal page that uses the `SSOUrl.getURL(companyId, userId)` method to dynamically construct a redirect URL to a landing page in the NetSuite application or web store. This URL should include all required parameters and any desired optional parameters. HTTP POST requests are not supported.

- Redirect the browser to the constructed URL.

**Note:** Using the Administrator role to log in to a web store is not supported.

For more information, see the following:

- Tables of Single Sign-on Redirect URL Parameters
- Example Single Sign-on Token and Redirect URLs

## Tables of Single Sign-on Redirect URL Parameters

The following tables describe the parameters used in single sign-on redirect URLs. HTTP POST requests are not supported.

Review all of the tables to ensure that you are including all of the parameters required for your purposes.

### Parameters used for both Application and Web Store Access

### NetSuite Application Access Only Parameters

### Web Store Access Only Parameters

In addition, see the following:

- For further insight into parameter usage, review the contents of the `SSOUrl.java` file.
- For example redirect URLs, see the help topic Example Single Sign-on Token and Redirect URLs.

### Parameters used for both Application and Web Store Access

The following table describes the parameters used for both the NetSuite application access (app) and web store access (site).
**Note:** Using the Administrator role to log in to a web store is not supported.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required/Optional</th>
<th>Programmatic Parameter</th>
<th>Command-Line Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>authentication token</td>
<td>string representing the encrypted token</td>
<td>Required</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic <em>Elements of the Authentication Token String.</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>See also <em>Example Inbound Single Sign-on Token.</em></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>partner ID</td>
<td>unique ID assigned by NetSuite for use with inbound single sign-on</td>
<td>Required</td>
<td>pid</td>
<td>p</td>
</tr>
<tr>
<td></td>
<td>this ID is associated with the public key you provided to NetSuite</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>landingurl</td>
<td>page that first displays for inbound single sign-on access, other than default</td>
<td>Optional</td>
<td>landingurl</td>
<td>l</td>
</tr>
<tr>
<td></td>
<td>Defaults are:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• for NetSuite application access, the user's home page</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• for web store access, the site home page</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The value of landingurl must be encoded, and after decoding it must be valid URL.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hideloginpage</td>
<td>Boolean indicating whether to hide the default inbound single sign-in login page from users and instead go to the page specified by the returnurl parameter</td>
<td>Optional</td>
<td>hideloginpage</td>
<td>h</td>
</tr>
<tr>
<td></td>
<td>By default, set to false. NetSuite recommends that it be set to true for web store access.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>returnurl</td>
<td>page where single sign-on users are redirected on session logout, timeout, and errors</td>
<td>Required if hideloginpage set to true</td>
<td>returnurl</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>Default is the inbound single sign-on login page.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Elements of the Authentication Token String

The format of the authentication token string prior to encryption is:

\(<companyID><space><userID><space><timestamp>\)

The companyID and userID elements represent the credentials used by the external application. (These credentials will be mapped to the NetSuite identity during inbound single sign-on.) Because spaces are used to delimit subtokens, companyID and userID elements cannot contain spaces.

See the following information about each element in the string:

- The companyID element is used by the external application to determine the company with which a user is associated, for example, ABCAutoParts. The companyID that you should use can vary. The goal is to ensure that the application token string is unique.
  - If you are a partner building an application for another company, the companyID should be a unique identifier of that company. You could use the company’s name, or any identifier you use to identify that company.
  - If you are building an integration for your own company, use your company name.
  - In any case, you can always use the NetSuite account ID as the companyID. To locate the account ID, go to Setup > Company > Setup Tasks > Company Information. The account ID field is located near the bottom of the right column.

- The userID string used by the external application as a user identifier, for example, John.Smith. It cannot contain spaces.

- The timestamp string is a decimal representation of the number of milliseconds since January 1, 1970, 00:00:00 GMT. The token is valid for 15 minutes after the timestamp contained in it.

NetSuite Application Access Only Parameters

The following table describes the NetSuite application access only parameters.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required/Optional</th>
<th>Programmatic Parameter</th>
<th>Command-Line Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>partner account/remote</td>
<td>External application-assigned ID for your company.</td>
<td>Required if target is app</td>
<td>pacct</td>
<td>rc</td>
</tr>
<tr>
<td>company ID</td>
<td>This value is identical to the companyID value used in the token. For more information about the companyID, see the help topic Elements of the Authentication Token String.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>partner user ID/remote</td>
<td>External application-assigned ID for your user.</td>
<td>Required if target is app</td>
<td>puid</td>
<td>ru</td>
</tr>
<tr>
<td>user ID</td>
<td>This value is identical to the userID value used in the token.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>application domain</td>
<td>Allows specification of domain and data center, for example, system.na1.netsuite.com.</td>
<td>Optional</td>
<td>ad</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If specified, the application domain for the base URL will</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Setting Up Inbound Single Sign-on

Web Store Access Only Parameters

The following table describes the web store only access parameters used in single sign-on redirect URLs. These parameters are used when the target is site. HTTP POST requests are not supported.

⚠️ **Important:** Always specify the NetSuite-hosted checkout domains (including the correct data center) in addition to the c parameter for faster routing if the performance of the login operation is a concern.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required/Optional</th>
<th>Programmatic Parameter</th>
<th>Command-Line Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>Your custom checkout domain, for example: checkout.mycompany.com. If you use a NetSuite-hosted checkout domain, enter the domain. The domain must include the correct data center identifier. See the help topic Checkout Domains for Data Centers.</td>
<td>Required for custom checkout domains. Recommended for NetSuite-hosted checkout domains.</td>
<td></td>
<td>d</td>
</tr>
<tr>
<td>NetSuite company ID</td>
<td>NetSuite-assigned account ID for your company</td>
<td>Required for NetSuite-hosted checkout domains.</td>
<td></td>
<td>c</td>
</tr>
<tr>
<td>site ID</td>
<td>internal ID on a NetSuite website record; distinguishes among multiple sites in your web store</td>
<td>Required when the c parameter is used.</td>
<td></td>
<td>n</td>
</tr>
<tr>
<td></td>
<td>integer displayed on the Web Site Preview page, as shown in Finding the Site ID Parameter</td>
<td></td>
<td></td>
<td>s</td>
</tr>
<tr>
<td></td>
<td>A site ID of 1 is valid even with only a single site.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Using the Administrator role to log in to a web store is not supported.

Checkout Domains for Data Centers

If you use a NetSuite-hosted checkout domain, for the domain parameter, use the correct data center identifier in your URL.

- NA West: checkout.netsuite.com
- NA East: checkout.na1.netsuite.com
- NA Northwest: checkout.na2.netsuite.com
- NA Central: checkout.na3.netsuite.com
- EU West: checkout.eu1.netsuite.com
- EU Central: checkout.eu2.netsuite.com
Finding the Site ID Parameter

You can find the value of the site ID parameter to set for a multi-site environment at Setup > Site Builder > Preview Web Site:

Note: A site ID of 1 is valid for a single site as well as for a multi-site environment.

Example Single Sign-on Token and Redirect URLs

Review the following examples to get a better understanding of inbound single sign-on tokens and redirect URLs:

- Example Inbound Single Sign-on Token
- Example Redirect URL for the NetSuite Application
- Example Redirect URL for the Web Store
- Example Redirect URL for Intermediate Third-party Login to Web Store

For details about the parameters used in redirect URLs, see the help topic Tables of Single Sign-on Redirect URL Parameters.

Example Inbound Single Sign-on Token

The following example illustrates the three stages of generating an inbound single sign-on token. The token is created with:

- the external application-assigned remote company ID, and
- the remote user ID for the user, and
- the current timestamp.

The token is then encrypted using a private key, and then hex-encoded so it can be passed as a redirect URL parameter.

Note: The hex-encoded, encrypted token string that is used as the URL parameter.

Stages of Token Generation

1. Plain Text String:
   ABCAutoParts John.Smith 1225479286770

2. Encrypted String:

3. Hex-Encoded, Encrypted String:
Setting Up Inbound Single Sign-on

Example Redirect URL for the NetSuite Application

The following is an example redirect URL for inbound single sign-on access to the NetSuite application:

```
https://system.netsuite.com/app/login/secure/sso.nl?
pid=198765&pacct=ABCAutoParts&puid=John.Smith&a=57E1A7DA1CD637E6BD1F6D7AF3F9EE9680DE0D3376786068E4448760CEC5D249031CA0B4618EB50C731703DE27150F663C7AC11D384CEBA43290B2EB58F1E1
```

The base URL in the redirect URL is determined by the target set in integration code; in this case, the target is set to `app`.

Values for the URL parameters in this example also are set in integration code: the partner ID (`pid`), the remote company id (`pacct`), the remote user ID (`puid`), and the hex-encoded encrypted token string (`a`).

⚠️ **Important:** The parameters listed above are valid parameters for this use case. For more information, see the Tables of Single Sign-on Redirect URL Parameters.

Do not use the `ck` and `cktime` parameters described in the Example Redirect URL for Intermediate Third-party Login to Web Store.

Example Redirect URL for the Web Store

Note: Using the Administrator role to log in to a web store is not supported.

The following is an example redirect URL for inbound single sign-on access to the web store:

```
https://checkout.netsuite.com/app/site/backend/sitesso.nl?
a=57E1A7DA1CD637E6BD1F6D7AF3F9EE9680DE0D3376786068E4448760CEC5D249031CA0B4618EB50C731703DE27150F663C7AC11D384CEBA43290B2EB58F1E1
www.abcautoparts.com/&c=198765&n=1
```

The base URL in the redirect URL is determined by the target set in integration code; in this case, the target is set to `site`.

Values for the URL parameters in this example also are set in integration code: the hex-encoded encrypted token string (`a`), the partner ID (`pid`), the hide login page indicator (`hideloginpage`), the return URL (`returnurl`), the NetSuite-assigned company ID (`c`), and the site ID (`n`).

⚠️ **Important:** The parameters listed above are valid parameters for this use case. For more information, see the Tables of Single Sign-on Redirect URL Parameters.

Do not use the `ck` and `cktime` parameters described in the Example Redirect URL for Intermediate Third-party Login to Web Store.

Example Redirect URL for Intermediate Third-party Login to Web Store

Note: Using the Administrator role to log in to a web store is not supported.

Inbound single sign-on supports the workflow where a customer visits a NetSuite shopping site, adds an item to the cart, clicks the Checkout button, and then is directed to a third-party site for login. As shown in the diagram below, after logging into the third-party site, the customer is directed to NetSuite checkout servers to complete a transaction.
To ensure that the shopping cart contents persist to the NetSuite checkout servers, parameters that allow the checkout servers to determine the original session must be included in the single sign-on call to NetSuite.

**Important:** The `ck` and `cktime` parameters described in the following procedure should only be used in situations when there is an intermediate third-party login required before proceeding to the Web Store.

**To ensure synchronization between the NetSuite web store checkout server and the shopping server:**

1. Include two additional parameters, `ck` and `cktime`, in the customized checkout link pointing to third-party servers for login. You can include these parameters by using tags in the customization text.
   
   You might, for example, put the tags directly into the URL you are substituting for the checkout URL as:

   ```
   ...&ck=<_NLSHOPPERID_>&cktime=<_NLCOOKIETIMESTAMP_>...
   ```

2. Upon receiving these parameters on the third-party login resource, read them, and then save them for addition to the URL to which the customer is redirected for single sign-on after login at the third-party site.

   Example:

   ```
   https://checkout.netsuite.com/app/site/backend/sitesso.nl?
   landingurl=http%3A%2F%2Fshopping.f.netsuite.com%2Fs.nl%3Fc%3D1035737&pid=1&c=1035737&a=792C4B61EF9BE695E9E9375FD78D24F25200EDEEEF01A416B03A2AAC41EE02C31F4580D33F0E7FED1C154BF055987AC9D8E189D01B9024B3A7F6E77C82B09F1FB99B0837BA0642CB58EA8B9839308503DF3ADDE3DD3F22ED37704D7C30171871C6439E0F69BCA49C6DAA2B5B1D2651490B6FA4E3FA4BB&ck=rBDDSzm-AdboaPPb&cktime=114233
   ```

   The base URL in the redirect URL is determined by the target set in integration code; in this case, the target is set to `site`.

   Values for the URL parameters in this example also are set in integration code: the page that first displays for inbound single sign-on access (`landingurl`), the partner ID (`pid`), the NetSuite-assigned company ID (`c`), the hex-encoded encrypted token string (`a`), the shopperid (`ck`), and a time stamp (`cktime`).
Setting Up Inbound Single Sign-on

Important: The parameters listed above are valid parameters for this use case. For more information, see the Tables of Single Sign-on Redirect URL Parameters. The ck and cktime parameters are additional parameters valid for this use case only.

SOAP Web Services Single Sign-on Operations

Important: As of 2018.1, new solutions using Inbound SSO for SOAP web services are not supported. Use Token-based Authentication (TBA) instead when creating new inbound SSO solutions for SOAP web services. See Token-based Authentication and Web Services.

The SOAP web services mapSso operation provides the ability to automate the mapping between users' external application credentials and NetSuite credentials.

- This operation provides inbound single sign-on access to NetSuite without users having to log in to NetSuite the first time this access occurs. Instead of the mapping between their external application credentials and NetSuite credentials being created at the time of this login, the mapping is created through SOAP web services.
- Use of this operation is required for inbound single sign-on access to the web store.

Note: Using the Administrator role to log in to a web store is not supported.

- This operation is applicable to accounts that have inbound single sign-on set up, and that have access to the associated external application credentials and encryption keys needed to generate the token.
- For more information, see the SOAP web services mapSso topic, which includes code samples.

Single sign-on mappings are not copied from a production account to a sandbox account when the sandbox is refreshed. These mappings must be recreated in the sandbox account for any users who require inbound single sign-on access to that account.

The SOAP web services login operation provides a mechanism for external applications to automate inbound single sign-on user login to NetSuite without the user's NetSuite credentials going through the external servers.

- This operation provides inbound single sign-on access to NetSuite without users having to click a link in the external application. The activities that occur when a user clicks this type of link instead occur behind the scenes.
- This operation is applicable to users who authenticate to NetSuite through the SOAP web services login operation; it is not applicable to users who authenticate to NetSuite by providing user credentials in the header of their SOAP requests.
- For more information, see the SOAP web services ssoLogin topic, which includes code samples.

Important: NetSuite hosts customer accounts in multiple data centers. For that reason, the correct URL for SOAP web services access varies depending on the data center hosting the account. Your integration must incorporate logic that dynamically determines the correct URL. With the 2012.2 and later endpoints, you should use the getDataCenterUrls operation to dynamically discover the correct URL. With older endpoints, you should use the REST roles service. For details, see the help topic The REST roles Service.

Error Handling for Inbound Single Sign-on

When an inbound single sign-on session is interrupted, NetSuite sends status codes to the external application. The external site can receive each of these status codes and display an appropriate error to the user.
The following status codes may be returned if the hideloginpage parameter is set to T (true): (If hideloginpage is set to F (false), the user is redirected to the login page.)

- **LOGIN_ERR_NO_MAPPING** - No SSO mapping of user authentication exists in NetSuite.
- **LOGIN_ERR_UNKNOWN** - Unexpected error occurred.
- **SESSION_TIMEOUT** - Session timed out in NetSuite.
  
  Note that each inbound single sign-on token includes a timestamp, and single sign-on access is only valid for 15 minutes.

The following status code may be returned independently of the hideloginpage parameter value:

- **LOGOUT** - User chose to log out.

### Setting Up a Single Sign-on Only Role

For security purposes, you can designate a NetSuite role as Single Sign-on Only. When a user logs in with a role that has been designated as Single Sign-on Only, validation is performed to ensure that the user is logging in through an inbound single sign-on mechanism. This mechanism can be either the NetSuite certificate-based inbound single sign-on feature or OpenID single sign-on feature.

The Single Sign-on Only role supports strict control of credentials from the external application. This type of role increases the security of an integrated application by prohibiting a SOAP web services or UI user from accessing the system with permissions and privileges that are specifically created for inbound single sign-on access only.

**Important:** You cannot use NetSuite for Outlook with a Single Sign-on Only role. Users who are not sure whether their role is compatible with NetSuite for Outlook should contact their account administrator.

**To designate a role as Single Sign-on Only:**

1. Go to Setup > Users/Roles > Manage Roles.
2. On the Manage Roles list page, select Edit or Customize next to the role you want to set as Single Sign-on Only.
3. Check the Single Sign-on Only Role box.
4. Click Save.

Next, assign this role to single sign-on users as needed.

For details about setting up roles in NetSuite, see the help topic Customizing or Creating NetSuite Roles.

### Mapping Users and Roles for Inbound Single Sign-on Access to NetSuite

NetSuite verifies a user's identity by comparing the remote system credentials passed in the token (company ID and user ID) to their NetSuite credentials (email, password, account, and role used to log in to NetSuite). To allow for this comparison and verification, the remote system credentials must be associated with, or mapped to, the NetSuite credentials. This mapping stores a permanent association between the user’s external application identity and their NetSuite identity.

The initial single sign-on account mapping must be to an Administrator role in NetSuite. This administrator mapping serves as authorization that NetSuite trusts the external authentication system.
This requirement gives NetSuite administrators control over when single sign-on functionality is available to their users. See the help topic Creating the Initial Mapping of the Administrator Role for Inbound Single Sign-on for more information.

**Note:** Using the Administrator role to log in to a web store is not supported.

After the initial mapping to the Administrator role is completed:

- For web store access, the administrator is required to use the SOAP web services mapSso operation to create the account mappings for multiple users so that they are available before users initiate single sign-on access.
  - If a user must access both Customer Center and non-Customer Center roles, they must have at least two mappings.
  - Separate mappings are required for each Customer Center role.
- For NetSuite access, the administrator can use the SOAP web services mapSso operation to create the account mappings for multiple users, or can instruct users to create their own mappings.

**Note:** The mapSso operation is a SOAP web services operation. To use SOAP web services, you must enable the feature in NetSuite. In addition, the role being mapped must include the SOAP web services permission set to Full. See Enabling the Web Services Feature and Assigning the Web Services Permission to a Role for more information.

If the administrator does not create mappings for users' external credentials and NetSuite credentials, users are required to create these mappings when they first log in with a role that requires single sign-on access. (This method of mapping is not supported for web store access.) See the help topic Creating Your Mapping for Inbound Single Sign-on to the NetSuite UI

Be aware of the following:

- If a NetSuite role used for inbound single sign-on access is deleted, the single sign-on mapping for any user with that role is automatically remapped to another role.
- If a user has a single sign-on mapping set up with a particular role and that role is removed from the user, the mapping is deleted. You can set up a new mapping for that user with a different role.
- There is no limit to the number of mappings you can create. The only limitation is that for each mapping the combination of the partner ID, partner account, and user id must be unique.
- Single sign-on mappings are not copied from a production account to a sandbox account when the sandbox is refreshed, or from one sandbox account to another. These mappings must be recreated in each sandbox account for any users who require inbound single sign-on access to that account.

**Note:** If a user requires single sign-on access for multiple accounts, you must use a different partner account-remote company ID for each single sign-on mapping for that user. For more information, see the help topic NetSuite Application Access Only Parameters.

- If a user must access both Customer Center and non-Customer Center roles, they must have at least two mappings.
- Separate mappings are required for each Customer Center role.

### Creating Your Mapping for Inbound Single Sign-on to the NetSuite UI

If your NetSuite administrator did not already create the mapping for you, the first time you log in from an external application to the NetSuite UI in your inbound single sign-on role, you must create a mapping.
Setting Up Inbound Single Sign-on

To create your mapping for inbound single sign-on to the NetSuite UI:

1. Log in to the external application to be used for inbound single-sign-on access to NetSuite.
2. Click the link to go to the NetSuite UI. The NetSuite Partner Login page displays.
3. On the NetSuite Partner Login page, enter your NetSuite email address and password.
4. Click Log in. The NetSuite Choose a role to create the mapping page displays.
5. Click the name of the role you will use for inbound single sign-on access to this account.

The mapping is now complete. You will automatically be logged in when accessing NetSuite by inbound single sign-on from your external application.

- If you must access both Customer Center and non-Customer Center roles, you must have at least two mappings.
- Separate mappings are required for each Customer Center role.

Technical Summary of Inbound Single Sign-on

In the following text, system A refers to an external application, and system B refers to NetSuite.

- HTTP POST requests are not supported.

This example discusses two systems, systems A and B, and a user that has identifiers ID_A and ID_B in the respective systems. In the absence of single sign-on, A and B would each require ID and PASSWORD presentation for user access. In order for system A to validate the user and then use single sign-on to redirect the user to system B, without requiring further user authentication, the following steps occur:

1. System A validates user ID_A as usual, requesting PASSWORD_A.
2. User works in system A and eventually clicks a link to system B.
3. System A creates a string T = ID_A + " " + TimeStampString.
   - ID_A is <companyID><space><userID>, so the entire token prior to encrypting and hex-encoding is <companyID><space><userID><space><timestamp>. The <companyID> in system A maps to a similar ID in system B.
   - Because a <space> is used to delimit subtokens in the token, none of the subtokens may contain <space> characters.
   - The timestamp string is a decimal representation of the number of milliseconds since January 1, 1970, 00:00:00 GMT.
4. System A encrypts T using RSA encryption with a private key, KA Pr, creating a token {T}KA Pr.
5. System A hex-encodes the encrypted bits so they can be transported as a URL parameter, the result being hex({T}KA Pr).
6. System A directs the user's browser to a landing link on system B, including hex({T}KA Pr) as a URL parameter.
7. System B hex decodes hex((T)KA pr ), yielding (T)KA pr.
8. System B uses the public key, KA pu, corresponding to KA pr to retrieve T from(T)KA pr.
9. System B checks that T was recently generated by observing TimeStampString. This check reduces
    the risk of the token being used outside the context of single sign-on between A and B.
10. System B looks up ID B in a table that maps {A,ID A } ID B.
11. System B trusts system A’s authentication procedure, and therefore logs user ID B into system B
    transparently.
SAML Single Sign-on

SAML (Security Assertion Markup Language) is an XML-based standard that supports communication of user data among different enterprise applications, called service providers (SPs). An identity provider (IdP) makes security assertions consumed by other service providers. A single IdP can perform user authentication for many SPs. A particular SP and an IdP can establish a circle of trust by providing each other with metadata in an XML format defined by SAML specifications, so that the SP accepts users authenticated by the IdP.

The NetSuite SAML Single Sign-on feature is based on the Security Assertion Markup Language (SAML) v2.0 specifications. For information about these specifications, click here. Any SAML 2.0-compliant application can serve as the IdP for SAML access to NetSuite.

The SAML Single Sign-on (SSO) feature supports inbound single sign-on access to NetSuite using authentication from a third-party IdP. This feature allows users who have logged in to an external application to go directly to NetSuite. Users do not need to log in separately to NetSuite, because authentication from the same IdP is used for login to both the external application and NetSuite. A user who accesses NetSuite using SAML SSO is directed to their NetSuite Home page. NetSuite account administrators can use role-based permissions in NetSuite to control which users have SAML SSO access to NetSuite.

Note: SAML single sign-on access to NetSuite UI honors any IP address rules for your company, or IP address restrictions for your employees, that you may have created in your NetSuite account. IP address rules or restrictions do not apply for SAML access to web stores or websites.

Task List for SAML SSO Set Up

Setting up SAML SSO requires some back-and-forth between NetSuite and the IdP of your choice.

1. In the NetSuite application, perform preliminary setup: enable the feature, create roles and assign SSO permissions, and assign users to the roles.
2. Using the IdP of your choice: create your NetSuite Service Provider (SP) configuration. The procedure varies depending on the IdP you choose to use.
   
   Note: Some IDPs already have NetSuite listed among their out-of-the-box service providers, while others require that you configure the set up of NetSuite as new SAML Service Provider yourself.

3. In the NetSuite application, complete the SAML Setup page: create the configuration in your account for your IdP.

See the following sections for detailed information on each step:

- Complete Preliminary Steps in NetSuite for SAML SSO
- Configure NetSuite with Your Identity Provider
- Complete the SAML Setup Page

If you are interested in setting up SAML SSO access to your Commerce web store, familiarize yourself with the SAML SSO documentation in this section. Then, see the help topic SAML Single Sign-on Access to Web Store for more information.
Complete Preliminary Steps in NetSuite for SAML SSO

To get started with SAML Single Sign-on (SSO), some preliminary setup steps must be completed in your NetSuite account. The feature must be enabled, and roles must be customized and SAML permissions added to those roles. SAML roles must be assigned to users.

The first step in setting up SAML SSO is to enable the feature, create roles and assign SSO permissions, and assign users to the roles.

See the following sections for detailed procedures:

- Enable the SAML Single Sign-on Feature
- Add SAML Single Sign-on Permissions to Roles
- Assign SAML Roles to Users
- Prepare to Provide NetSuite SP Metadata to Your IdP

Enable the SAML Single Sign-on Feature

To complete the following procedure, you must be logged in to NetSuite with an Administrator role or in another role that has the Enable Features permission.

To enable the SAML Single Sign-on feature:

1. Go to Setup > Company > Enable Features and click the SuiteCloud subtab.

2. Scroll down to the Single Sign-on section, and check the SAML Single Sign-on box. Agree to the SuiteCloud Terms of Service when prompted.

3. Click Save.
Complete Preliminary Steps in NetSuite for SAML SSO

Warning: By enabling the SAML Single Sign-on feature, you allow users to access and use your NetSuite account directly from a third-party service that may not have the same authentication and security features as NetSuite. This feature also extends NetSuite administration of user access to the administrators of the identity management system. You need to ensure that NetSuite account use through SAML meets all of your security, regulatory, and other compliance obligations, including Payment Card Industry (PCI) Data Security Standards.

Add SAML Single Sign-on Permissions to Roles

You might want to customize a standard NetSuite role (or roles) to for use with SAML Single Sign-on (SSO) permissions. You can also add SAML SSO permissions to existing roles assigned to users that require this type of access.

Note: If a role is already designated as two-factor authentication (2FA) required, and you add the SAML SSO permission to the role, the 2FA requirement will be ignored. The SAML SSO permission takes precedence.

To complete the following procedure, you must be logged in to NetSuite with an Administrator role. If you need more detailed information about creating roles in NetSuite, see the help topic Customizing or Creating NetSuite Roles.

To customize roles and add SAML permissions:

1. Go to Setup > Users/Roles > User Management > Manage Roles.
2. Choose a role and click Customize.
3. Create a unique and identifiable name for the role. For example, you could replace the word Customize in the role name with the word SAML.
4. Click the Permissions tab.
5. On the Setup subtab, select the appropriate SAML permission from the list, and click Add. There are two SAML permissions. Add one or both permissions to the role as appropriate. See SAML SSO Permissions.
6. Click Save.

For more information about SAML permissions, see the following:
- SAML SSO Permissions
- SAML SSO Access for Center Roles
- SAML SSO Permission Limitations

SAML SSO Access for Center Roles

You can add the SAML Single Sign-on permission to customized versions of the following center roles: Customer Center, Employee Center, Vendor Center, Partner Center, and Advanced Partner Center. Center roles are different from other NetSuite roles in that you can only add a limited set of permissions to them.

To add the SAML Single Sign-on permission to a customized center role:

Important: No special permission is required to grant a customer center role SAML access to a website. The SAML permission is enabled for all customer center users, after the SAML setup for the website is completed.

1. Go to Setup > Users/Roles > Manage Roles.
2. Click the **Edit** link for a customized center role or click the **Customize** link for a standard center role.

3. On the Role page, click the **Permissions** subtab.

4. On the **Setup** subtab, set the **Level** to **Full** for the SAML Single Sign-on permission.

### SAML SSO Permissions

When the SAML Single Sign-on feature is enabled, the following permissions are available:

- **Set Up SAML Single Sign-on** - permits users other than NetSuite account administrators to view and edit the SAML Setup page. (The Administrator role already has this permission.)

- **SAML Single Sign-on** - requires users to log in to the NetSuite UI using SAML SSO. This permission must be explicitly assigned to a role. However, a user assigned this permission will not be able to log in to the NetSuite UI from the standard login page with their username and password.

**Important:** No special permission is required to grant a customer center role SAML SSO access to a website. After the SAML setup for a website is completed, the SAML permission is automatically enabled for all customer center users.

Both of these permissions are Setup type permissions that support only a Full access level.

To provide SAML single sign-on access to users, the SAML Single Sign-on permission can be added to an existing role that is already assigned to users. Or, a new role can be created to which this permission can be added, and this new role can then be assigned to users.

**Important:** You cannot add SAML Single Sign-on permission to a role that has SuiteAnalytics Connect permission.

Permissions are added to roles on the Role record page, available at Setup > Users/Roles > Manage Roles.

**Important:** After the SAML Single Sign-on permission has been assigned to a role, there is a small delay before a user can use this role to log in using SAML single sign-on. This delay is related to caching; the new permission is not available until after the cache has timed out.

For more information about adding permissions to roles, see the help topic Customizing or Creating NetSuite Roles.
SAML SSO Permission Limitations

SAML Single Sign-on roles and permissions have various limitations that are intended to prevent problems.

For example, the NetSuite account administrator role does not have SAML Single Sign-on permission and no user can log in using SAML single sign-on as an administrator. This limitation ensures that an administrator can always log in and resolve any problems that might occur with the third-party IdP setup or SAML access.

Another example of a limitation is that account administrators cannot add SAML Single Sign-on permission to a role that has SuiteAnalytics Connect permission. SAML access is not supported for SuiteAnalytics Connect.

Some limitations are intended to ensure that the account administrator has absolute responsibility for explicitly deciding who is allowed to access their NetSuite account using SAML Single Sign-on. The account administrator is deciding to trust the third-party IdP to authenticate and allow access to their NetSuite account. This is the reason for the following limitations:

- A user who has accessed NetSuite with a role that does not have SAML Single Sign-on permission cannot access any roles that do have SAML Single Sign-on permission.
- As of 2018.1, it is up to an account administrator to decide whether users should be locked in a single account. See Account Attribute for more information. (In previous releases, a user who accessed NetSuite through SAML Single Sign-on could not access any roles that belonged to a different NetSuite account. SAML Single Sign-on access was provided to only a single account.)

Some limitations are intended to ensure there are no conflicts resulting from having two different trust authorities (the third-party IdP and NetSuite) authenticating a single user. After SAML is enabled for certain roles in an account, NetSuite trusts the third-party identity provider. This is the reason behind the following limitations:

- A user with a role that has SAML Single Sign-on permission cannot log in directly to the NetSuite user interface using the standard NetSuite login page.
- A user who has accessed NetSuite through SAML Single Sign-on cannot access any roles that do not have SAML Single Sign-on permission. This prevents users from switching from a SAML role to a non-SAML role with greater privileges.
- Only one type of inbound single sign-on permission can be assigned to a specific role. If a role has SAML Single Sign-on permission, it cannot have OpenID Single Sign-on permission or be granted inbound single sign-on access through the NetSuite Inbound Single Sign-on feature.

Assign SAML Roles to Users

To complete the following procedure, you must be logged in to NetSuite with an Administrator role. If you need more detailed information, see the help topic NetSuite Users Overview.

To assign a SAML Single Sign-on role to users:

Important: A user with a role that has SAML Single Sign-on permission cannot log in directly to the NetSuite user interface on the standard NetSuite login page with the SAML role.

The following procedure is for adding a role with the SAML Single Sign-on permission to users.
Complete Preliminary Steps in NetSuite for SAML SSO

1. Find the appropriate entity record for the user. Go to List > Employee > Employee.
2. Click the name of the user.
3. Click the Access subtab.

**Important:** No special permission is required to grant a customer center role SAML access to a website. The SAML permission is enabled for all customer center users, after the SAML setup for the website is completed.

4. Click Edit.
5. Select your custom SAML Single Sign-on role from the list.
6. Click Add.
7. Click Save.

Prepare to Provide NetSuite SP Metadata to Your IdP

After the SAML Single Sign-on feature is enabled, NetSuite account administrators and users with the Set Up SAML Single Sign-on permission can view and edit the SAML Setup page in NetSuite. How you configure NetSuite as a Service Provider (SP) with the Identity Provider (IdP) of your choice depends on the IdP you have selected. To prepare for any eventuality, before you attempt to set up SAML with your IdP, you should gather some information from the SAML Setup page in NetSuite.

**To copy the NetSuite SP metadata file and related URL:**

The person responsible for configuring SAML access to NetSuite on the IdP side should perform the following steps.

1. Go to Setup > Integration > SAML Single Sign-on.
2. Copy the URL shown in the NetSuite Service Provider Metadata field, and save it where you can retrieve it when necessary.
3. Download the SP metadata file to your computer. Remember the location you save the file to.
**Important:** The URL shown on the SAML Setup page in the NetSuite Service Provider Metadata field in the following screenshot is obscured, because the URL varies depending on the data center where your account is hosted.

**Note:** If your account has been or is moved to a different data center location, information in the NetSuite SP metadata file will also change. You must provide the information in the SP metadata file to your IdP from the current location of your account.

Configure NetSuite with Your Identity Provider

It is not possible to provide detailed instructions for configuring NetSuite as a Service Provider (SP) with your Identity Provider (IdP). Refer to the documentation available from your IdP for configuring SAML access. However, see the following procedure for basic guidance on what must be accomplished to set up SAML access to NetSuite with your IdP. The exact steps will vary, depending on your IdP. The procedure will also vary depending on whether the NetSuite application is already configured by your IdP, or if you must create the NetSuite application yourself with your IdP.
Configure NetSuite with Your Identity Provider

Note: Your IdP could be a web application or an on-premises solution. The NetSuite application could already be included in their list of SP applications. The IdP might have a setup wizard or a manual to guide you through the process.

To configure SAML with your IdP:

1. Go to your IdP website or an on-premises administration console, and follow the application setup instructions from your IdP.

   Note: You must create a new SP application for NetSuite. Refer to your IdP's documentation for directions on how to do this.

2. Provide the NetSuite Service Provider Metadata to your IdP by one of the following methods:
   a. Upload the NetSuite SP metadata file, or:
   b. Paste the URL for the NetSuite SP metadata file in the appropriate field with your IdP, or:
   c. Manually configure SAML on the IdP side by copying information from specific fields in the NetSuite Service Provider Metadata file to the IdP.

   If you need instructions because you must manually upload a certificate file, see Extract an Encryption Certificate or Signing Certificate from the SP Metadata File.

Your IdP (website or on-premises console) From the NetSuite Service Provider Metadata file

<table>
<thead>
<tr>
<th>SP Entity ID</th>
<th>Always refer to the NetSuite Service Provider Metadata file in your account.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy the SP entityID from the NetSuite Service Provider metadata file you downloaded from the SAML Setup page in your account.</td>
<td></td>
</tr>
<tr>
<td>The SP entityID is shown in the first line of the file.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Assertion Consumer Service</th>
<th>Always refer to the NetSuite Service Provider Metadata file in your account.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy the URL from the NetSuite Service Provider metadata file you downloaded from the SAML Setup page in your account.</td>
<td></td>
</tr>
<tr>
<td>The value varies depending on the data center where your account is hosted, and on the account type. There are many AssertionConsumerService lines for Location in the file. Use the URL in your file that includes isDefault=&quot;true&quot; in the line.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Single Logout Service</th>
<th>Always refer to the NetSuite Service Provider Metadata file in your account.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy the URL from the NetSuite Service Provider metadata file you downloaded from the SAML Setup page in your account.</td>
<td></td>
</tr>
<tr>
<td>The value varies depending on the data center where your account is hosted, and on the account type.</td>
<td></td>
</tr>
<tr>
<td>Ensure you use a POST binding.</td>
<td></td>
</tr>
</tbody>
</table>

3. Your IdP also has an IdP metadata configuration file. You must copy the URL for this file, or download the IdP metadata file. (Later, you must either enter the URL or upload the file into NetSuite on the SAML Setup page.)

4. With your IdP, you must assign (or provision) the NetSuite application to the SAML users in your account.

In many cases, the previous steps take care of all the information you need to provide to the IdP. For more information about signing assertions, encryption, and SAML attributes, see IdP Metadata and SAML Attributes.
Complete the SAML Setup Page

When the SAML Single Sign-on feature is enabled, the SAML Setup page is available at Setup > Integration > SAML Single Sign-on, to NetSuite account administrators and to users with the Set Up SAML Single Sign-on permission. (For details about SAML Single Sign-on permissions, see Add SAML Single Sign-on Permissions to Roles.)

Important: The URL link to the NetSuite Service Provider Metadata field in the following screenshot is obscured, because the URL varies depending on the account type and your data center location.

Note: If your account is moved to a different data center, information in the NetSuite SP metadata file will also change. You must provide the information in the SP metadata file to your IdP from the current location of your account. For instructions, see Prepare to Provide NetSuite SP Metadata to Your IdP and Configure NetSuite with Your Identity Provider.

For details about completing the SAML Setup page, see:

- Defining the NetSuite Configuration for SAML
- Set Up Your Identity Provider (IdP) in NetSuite
Defining the NetSuite Configuration for SAML

To support SAML single sign-on access to NetSuite, you must define the following on the SAML Setup page:

- The Logout Landing Page.
- Optionally, the Primary Authentication Method.

Logout Landing Page

Logout Landing Page - after logging in to NetSuite through SAML single sign-on, this is the URL for a page that users should be redirected to when they log out of NetSuite. An IdP Single Logout page can be specified for Single Logout to work.

Note: This solution is not part of the SAML 2.0 standard. There is no guarantee that this will work.

Primary Authentication Method

The Primary Authentication Method is optional.

- By default, the Primary Authentication Method option is not checked. If SAML users click a link to access NetSuite when no active NetSuite session exists, they are redirected to the NetSuite login page. This redirect might cause issues for users who do not know their NetSuite credentials.
- If you check the Primary Authentication Method box, users can be redirected to the external IdP login page. This redirect is available if:
  - the user has already been logged in, the redirect occurs based on previous experience with NetSuite.
  - the access link includes the NetSuite account ID set as the c or compid URL parameter, formatted like the following:
    https://system.netsuite.com/app/center/card.nl?c=<ACCOUNTID>
Set Up Your Identity Provider (IdP) in NetSuite

SAML single sign-on access to NetSuite requires that you specify an XML file that defines the identity provider to be used for authentication and includes required metadata for this identity provider. The format of this file must be aligned with SAML v2.0 specifications.

On the SAML Setup page, the IdP metadata file can be specified by entering a URL or by uploading the actual XML file. This is the information you gathered when you were setting up NetSuite with your IdP.

You must do one of the following:

- Choose **Indicate IDP metadata URL** and enter the location URL of the metadata file.
- Or, choose the **Upload IDP metadata File** option and browse to locate the file.

**Note:** If you need to make changes to the IdP configuration, see Update Identity Provider Information in NetSuite.

Update Identity Provider Information in NetSuite

After you have defined an identity provider for SAML Single Sign-on access, you can make changes as needed to the identity provider configuration on the SAML Setup page. Actions you can take include:

- **Update the IdP Configuration File**
- **Remove the Current IdP Metadata**
- **Change Your IdP for NetSuite**

Update the IdP Configuration File

Complete the following procedure to update the IdP configuration file. Updating the IdP configuration file could be necessary, for example, if the existing file in NetSuite contains expired meta information.
To update the IdP configuration file:

1. Log in to the website of your IdP
2. Locate the IdP metadata configuration file for the NetSuite application.
3. Copy the URL for this file or download the IdP metadata file from your IdP and remember the downloaded location.
4. Go to Setup > Integration > SAML Single Sign-on in your NetSuite account.
5. Under the Update Identity Provider section of the SAML Setup page, the new IdP metadata file can be specified in NetSuite by either:
   a. Entering the URL in the Indicate IDP Metadata URL field, or:
   b. Select Upload IDP Metadata File and click Choose File. Navigate to the location of the IdP configuration file you downloaded, select the file, and click Open.
6. Click Submit.

**Important:** If your company uses SAML SSO in multiple accounts with a shared configuration, repeat this procedure for all your accounts. Upload the same IdP metadata file in all the NetSuite accounts that you want to use with SAML SSO.

Remove the Current IdP Metadata

You can remove the current identity provider metadata without replacing it with another identity provider by clicking the button.

**Important:** This procedure removes the current IdP metadata from your NetSuite account, deletes the information in the Logout Landing Page field, and clears the Primary Authentication Method box.

To remove the current IdP metadata

1. Go to Setup > Integration > SAML Single Sign-on in your NetSuite account.
2. Under Actions, click Delete IDP Configuration.
Change Your IdP for NetSuite

You can change your current identity provider entering a URL or uploading an XML file that contains the metadata for a different identity provider.

To change your IdP

1. Log in to the website of your new IdP.
2. Locate the IdP metadata configuration file for the NetSuite application.
3. Copy the URL for this file or download the IdP metadata file from your IdP and remember the downloaded location.
4. Go to Setup > Integration > SAML Single Sign-on in your NetSuite account.
5. Under the Update Identity Provider section of the SAML Setup page, the new IdP metadata file can be specified in NetSuite by either:
   a. Entering the URL in the Indicate IDP Metadata URL field, or:
   b. Select Upload IDP Metadata File and click Choose File. Navigate to the location of the IdP configuration file you downloaded, select the file, and click Open.

IdP Metadata and SAML Attributes

See the following for more information about IdP metadata and specifying SAML attributes.

- IdP Requirements
- NameID and Email Attributes
- SAML Response Example
IdP Requirements

The Identity Provider metadata file should map required attributes between the identity provider and NetSuite, so that NetSuite can accept the identity provider’s SAML assertions.

See the following:

- Supported Encryption and Signature Options
- Extract an Encryption Certificate or Signing Certificate from the SP Metadata File
- Mapping of SAML Attributes
- SAML Attribute Statements
- SAML Response Example

Supported Encryption and Signature Options

At a minimum, NetSuite requires that an assertion be signed. Also, on the IdP side, an administrator can opt for several different levels of encryption.

NetSuite supports the following levels of encryption:

- The whole assertion can be encrypted.
- All attributes and NameID can be encrypted.
- Only the attributes can be encrypted.
- Only the NameID can be encrypted.

Extract an Encryption Certificate or Signing Certificate from the SP Metadata File

Use the following procedure if you must extract the encryption or signing certificate from the NetSuite Service Provider Metadata file. A Signing Certificate is only required if you are using an SP-initiated flow, or if you are using Single Logout (SLO).

To extract a certificate from the SP metadata file:

1. Download the SP metadata file from your NetSuite account.
   a. Go to Setup > Integration > SAML Single Sign-on.
   b. Download the SP metadata file to your computer. Remember the location you save the file to.
2. Create a new file in a text editor and enter the following text exactly as shown:
   
   ```
   -----BEGIN CERTIFICATE-----
   -----END CERTIFICATE-----
   ```
3. Use a text editor to open the SP metadata file you saved to your computer.
4. Copy the appropriate line from the SP metadata file.
5. Paste the line you copied from the SP metadata file to the blank line between the -----BEGIN CERTIFICATE----- and -----END CERTIFICATE----- lines.

6. Save the PEM-encoded file.

7. Follow your IdP's documentation for providing the certificate file to your IdP (for example, upload the file, or paste the content of the file into a provided form.)

### Mapping of SAML Attributes

See the following table for a mapping of SAML attributes to NetSuite parameters, and whether they are required or optional.

<table>
<thead>
<tr>
<th>SAML Attribute</th>
<th>NetSuite Parameter</th>
<th>Required or Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>account</td>
<td>accountID</td>
<td>Optional, unless:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- you are sending the role attribute.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- you are sending the site attribute.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- access to both non-customer center and customer center SAML roles is needed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sending the account attribute locks user access to a single account.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See Account Attribute for more information.</td>
</tr>
<tr>
<td>role</td>
<td>role</td>
<td>Optional.</td>
</tr>
<tr>
<td>site</td>
<td>site ID</td>
<td>Required for web store access.</td>
</tr>
<tr>
<td>NameID or email</td>
<td>user email address</td>
<td>Required, must use the NameID attribute or the email attribute.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>See NameID and Email Attributes for more information.</td>
</tr>
</tbody>
</table>

### SAML Attribute Statements

See the following sections for more information about SAML attributes.
IdP Metadata and SAML Attributes

- Account Attribute
- Role Attribute
- Site Attribute
- NameID and Email Attributes
  - Supported NameID Formats

**Account Attribute**

The account attribute is your NetSuite account ID. If you do not know your NetSuite account ID, a user with an administrator role can go to Setup > Company > Company Information to view the Account ID field. The account attribute is optional, unless:

- If you are sending the role attribute, then account is required.
- If you are sending the site attribute, then account is required.
- If users need access to both their non-customer center and customer center SAML roles, then account is required.

⚠️ **Important:** If you send the account attribute, users are locked into a single company account, and will not be able to switch between multiple accounts that trust the same IdP.

**Role Attribute**

The ability to define a role ID is particularly useful if you have a SuiteCommerce website. It is not possible for a user to switch roles when logged in to a website. With the role attribute, you can define the SAML role to be used for login. The role defined in the assertion is treated as a default role.

The role attribute can be passed along with the SAML assertion as an additional attribute. If the role attribute is sent, the assertion must also include the account attribute.

**Site Attribute**

Setting the site attribute (the site ID) is required for web store access. If you are sending the site attribute, you must also set the account attribute. If you do not know the site ID, see the help topic Finding the Site ID Parameter.

ℹ️ **Note:** When the site attribute is provided, the user is directed to web store with corresponding site id. It is not possible to route the SAML login to either the NetSuite account or to a web store based on the role in which the user logs in to the IdP.

**NameID and Email Attributes**

The user email is required. It must be provided either as the value in the NameID attribute or the email attribute.

ℹ️ **Note:** If using both the NameID and the email attributes, the values for these attributes must be identical, unless you are using the transient format. If NetSuite receives a SAML Assertion with a transient NameID, it must also contain an email attribute statement with the user email address. The values in transient NameID tag and email attribute statement do not need to be identical.

**Supported NameID Formats**

The following formats are supported for the NameID attribute:
IdP Metadata and SAML Attributes

- emailAddress
- transient
- unspecified

⚠️ Important: No matter which of these formats you choose to use, the NameID value must contain an email address.

The SAML Response Example illustrates the use of the emailAddress format for NameID.

The following line indicates use of the unspecified format:

```xml
<saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">jsmith@example.com</saml2:NameID>
```

SAML Response Example

The following is an example of a SAML Response, showing parts of the SAML assertion element. If you do not provide the required attributes in your file, you receive error messages, for example: Email must be provided using NameID value or the email attribute.

The following example illustrates one way to provide these values:

```xml
... 
<saml:Subject>
  <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress" SPNameQualifier="http://www.netsuite.com/sp" jsmith@example.com</saml:NameID>
  ...

  <saml:AttributeStatement>
    <saml:Attribute Name="email">
      <saml:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">jsmith@example.com</saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute Name="account">
    </saml:Attribute>
    <saml:Attribute Name="role">
    </saml:Attribute>
    <saml:Attribute Name="site">
    </saml:Attribute>
  </saml:AttributeStatement>
  ...
```

Interactions with NetSuite Using SAML

SP-initiated and IdP-initiated Flows

There are two possible single sign-on flows, or authentication flows, in the SAML 2.0 standard: SP-initiated and IdP-initiated. NetSuite supports both types of flows.
Interactions with NetSuite Using SAML

**Note:** Only an IdP-initiated flow is supported for SAML access to a Commerce web store. For more information, see the help topic SAML Single Sign-on Access to Web Store.

The SP-initiated Flow

To trigger an SP-initiated flow:

- SAML must be set as a primary authentication method, or:
- A user should have a browsing history using SAML, and a deep link should be used to trigger the flow.

For more information, see the Primary Authentication Method in Defining the NetSuite Configuration for SAML.

**SAMLRequest and RelayState**

To initiate the login protocol exchange, the SAMLRequest must be in an SP-initiated flow. RelayState is an optional parameter to preserve and convey state information that is transferred with the SAMLRequest message. For detailed information, refer to the SAML 2.0 specification. Go to [https://www.oasis-open.org/standards/samlv2.0](https://www.oasis-open.org/standards/samlv2.0).

You can configure the value of the RelayState attribute on the IdP side. However, for security reasons, NetSuite does not support redirects to external pages (other than NetSuite pages) through RelayState attribute in the SAML assertion.

Single Logout (SLO)

NetSuite has limited support for Single Logout (SLO) functionality. IdP-initiated SLO is supported for the NetSuite UI. The following is not supported:

- IdP-initiated SLO is not supported for SuiteCommerce web stores.
- SP-initiated SLO is not supported for the NetSuite UI or for SuiteCommerce webstores.

**Note:** The following solution is not part of the SAML 2.0 standard. If SP-initiated SLO is desired, and if your IdP supports this functionality, you could enter the Single Logout Service URL of your IdP in the Logout Landing Page field. There is no guarantee that this will work, as it depends on how your IdP implemented and supports the SAML SLO functionality.

SAML SSO in Multiple NetSuite Account Types

If you are using SAML Single Sign-on (SSO) in more than one account type, be aware of the following information.

Set Up and Configure SAML SSO in More Than One Account

The Shared Identity Provider (IdP) feature in 2018.1 introduced the possibility to trust the same IdP from multiple NetSuite accounts.

This list details four important changes as a result of the Shared IdP feature:

1. There is no longer a unique constraint on the IdP entity ID in NetSuite.
2. Users can log in and switch between NetSuite accounts trusting the same IdP.
3. Administrators are no longer required to create independent service provider (SP) configurations on the IdP side for every NetSuite account.
4. Only one NetSuite SP configuration is required, which removes problems that may have been encountered due to IdPs requiring unique SP entity IDs.

You can use the same IdP metadata file for all your NetSuite account types: for example, your production, sandbox, and Release Preview accounts. However, your SAML configuration is not copied from your production account to other account types.

- **Sandbox:** You must configure SAML in your sandbox account after each refresh.
- **Release Preview:** You must configure SAML in your Release Preview account when it becomes available before each new NetSuite release.

**Important:** Some IdPs require a unique entity ID for every SP. If this is the case with your IdP, follow the procedure below to so that there is a single SP configuration for all NetSuite accounts.

---

**Enable SAML in Multiple NetSuite Account Types**

The following procedure does not contain all the details for setting up and configuring SAML. For more details on each step, see the following topics:

- Complete Preliminary Steps in NetSuite for SAML SSO
- Configure NetSuite with Your Identity Provider
- Complete the SAML Setup Page

**To set up and configure a single IdP in multiple accounts:**

Complete the following procedure to use the same IdP in multiple NetSuite account types (for example, your production account and a sandbox account. You must be a NetSuite account administrator or a user with the Set Up SAML Single Sign-on permission to access the SAML Setup page.

1. Go to Setup > Integration > SAML Single Sign-on in one of your NetSuite accounts (production, sandbox, or Release Preview account.)
2. Click the link below the field labeled **NetSuite Service Provider Metadata** to download the SP metadata file.
3. Following the instructions provided by your IdP, create a single SP configuration, and upload the NetSuite SP metadata file.

**Important:** Do not specify an account parameter on the identity provider's side.

4. Download the IdP metadata from your IdP.
5. Upload the same IdP metadata file in all of the NetSuite accounts that you want to use with SAML.

**Note:** It is also possible to use the Shared IdP feature for multiple accounts of the same type. For example, you can use a single IdP for two or more production accounts, or several sandbox accounts.

---

**NetSuite SAML Certificate References**

The NetSuite SAML certificate is referenced in the NetSuite Service Provider metadata and in the SAML identity provider (IdP) metadata. This certificate is valid for a designated period of time, usually a number
of years. As the certificate expiration date approaches, NetSuite will renew it. After the renewed certificate is available, the NetSuite service provider metadata file will be automatically updated to include data from the renewed certificate. The NetSuite SAML Setup page, at Setup > Integration > SAML Single-Sign-on, provides a link that can be clicked to view the contents of this file. Certificate references in IdP metadata may not be automatically updated. Account administrators will need to review certificate references in IdP metadata, and manually update them as necessary, to ensure they point to the renewed certificate. NetSuite Customer Support will provide advance notice of SAML certificate expiration to affected customers.

**Note:** For information about removing SAML access to NetSuite after the SAML Setup page has been completed, see Remove SAML Access to NetSuite.

### Remove SAML Access to NetSuite

There are multiple ways to remove SAML single sign-on access to NetSuite.

- Either of the following actions removes SAML access to NetSuite for a user or group of users in your account:
  - Removing the SAML Single Sign-on permission from the users' roles.
  - In extreme cases, editing the users' employee records in NetSuite to make them inactive.
- The following actions remove SAML access to NetSuite for all users in your account:
  - Ensure that SAML roles are either inactivated, or SAML permissions are removed from the roles.
  - Disabling the SAML Single Sign-on feature in NetSuite.

**Note:** You can remove identity provider metadata on the SAML Setup page. See Update Identity Provider Information in NetSuite for information on viewing or removing the identity provider metadata for SAML access to web stores, see the help topic SAML Single Sign-on Access to Web Store.

### FAQ: SAML SSO

The following section contains answers to questions about setting up and using SAML SSO with NetSuite. SAML SSO in NetSuite is based on the Security Assertion Markup Language (SAML) v2.0 specifications. See OASIS Security Services (SAML) TC for a link to the SAML specifications. The complete SAML v2.0 OASIS Standard set (PDF format) and schema files are available in a .zip file. See also SAML Single Sign-on for information about setting up SAML in NetSuite.

### SAML SSO and Sandbox Accounts

**When I access my NetSuite production account through SAML SSO, can I switch roles to access my SAML role in a sandbox account?**

It depends on how your SAML is set up, whether or not the account ID is specified. See SAML SSO in Multiple NetSuite Account Types for more information.

**When I access my NetSuite production account through SAML SSO, can I switch roles to access my non-SAML roles in my production or sandbox accounts?**

No. It is not possible to access SAML roles and non-SAML roles in the same session.
When I log in to my NetSuite production account in a non-SAML role, can I switch roles to other non-SAML roles in my production or sandbox accounts?

Yes.

Technical Questions about SAML

Is encryption required?

As stated in the NetSuite Service Provider (SP) metadata, encryption is not required. At minimum, it is required only that assertions be signed (WantAssertionsSigned="true"). But an identity provider (IdP) can set a higher level of security using encryption. Refer to the SAML specifications to learn more about the encryption options SAML supports.

What Secure Hash Algorithm (SHA) is supported, SHA1 or SHA256?

The answer to this question is tied to the SAML 2.0 protocol. SAML relies on the XML-Signature Syntax and Processing specification (D. Eastlake et al. XML-Signature Syntax and Processing. World Wide Web Consortium, February 2002.) For more information, see http://www.w3.org/TR/xmldsig-core/. The only supported hashing function in this specification is SHA1. The recommended signature method is RSAwithSHA1.

What bindings are supported?

NetSuite does not support non-secure bindings. All of the bindings require TLS. Our Assertion Consumer Service only accepts the HTTP-POST binding. This is described in the Service Provider Metadata file. To view the NetSuite SP metadata file in your account, see Prepare to Provide NetSuite SP Metadata to Your IdP.

Do all SAML 2.0 messages have authenticity and integrity protection using a digital certificate?

The whole assertion message must be signed by the IDP private key and sent over HTTPS. NetSuite only supports use of the TLS 1.2 protocol for secure communication.

Does the Response for any message that does not have authenticity and integrity protection always indicate failure?

Yes, it does. At a minimum, NetSuite requires that an assertion be signed.

If a message or elements of a message are digitally signed, does the relying party always validate the public key of the digital signature?

Yes, it does.

Are there any revocation checks done against the signature (such as CRL or OCSP)?

There are no automatic checks. The revocation must be done by an account administrator, by removing an IDP’s metadata from the NetSuite account settings.

Are all SAML 2.0 messages sent through an HTTP binding using the Transport Layer Security (TLS) protocol?

NetSuite only accepts requests sent through HTTPS (TLS). NetSuite only supports use of the TLS 1.2 protocol for secure communication.
Does the Service Provider process the InResponseTo attribute of the Response to ensure the Response was intended for them and is still fresh?

For the SP-initiated flow, this check is included as per the SAML standard. Both IdP-initiated and SP-initiated flows are supported. See Interactions with NetSuite Using SAML for more information.

Does the Service Provider process the Destination attribute of the Response to ensure the Response was intended for them?

Yes, it does.

Does the Service Provider process the SubjectConfirmationData element to ensure the Assertion was intended for them?

Yes, it does.

Does the Service Provider validate the NotOnOrAfter attribute of the Conditions element to ensure the Assertion is still fresh?

Yes, it does.

Does the Service Provider process the AudienceRestrictions element to ensure the assertion was intended for them?

Yes, it does.

Does the Service Provider process the AuthnContext element to ensure class of Authentication?

Yes, it does.
OpenID Single Sign-on

The OpenID Single Sign-on feature in NetSuite supports inbound single sign-on to NetSuite from G Suite (formerly known as Google Apps for Work). This feature allows users who have logged in to G Suite to go directly to NetSuite. Users do not need to log in separately to NetSuite, because their Google identity is used to access their NetSuite data. NetSuite account administrators can control OpenID access through role-based permissions in NetSuite.

**Note:** As an alternative to OpenID, SAML 2.0 is available for single sign-on access to NetSuite. G Suite supports SAML 2.0. For more information, see SAML Single Sign-on. Also, the G Suite Administrator Help website provides documentation on setup and configuration of SAML SSO to NetSuite. Click the following to view Google's documentation: https://support.google.com/a/answer/6194968?hl=en. After the setup and configuration is completed, your users can use their G Suite credentials to access NetSuite.

The OpenID Single Sign-on feature in NetSuite uses Google's OpenID Connect (OAuth 2.0 for login). OpenID Connect is a federated authentication protocol that allows single sign-on from an identity provider to applications that are set up as relying parties (or service providers). Single sign-on allows users to log in a single time, to the identity provider, and then move to other applications without needing to log in again.

**Note:** OpenID Single Sign-on access to NetSuite is only supported for UI to UI integrations. It is not supported for access to a web store, or for access to NetSuite through SOAP web services. OpenID Single Sign-on access is not supported for customer center roles.

The following steps are required to implement OpenID access to a NetSuite account:

1. Your organization must have previously set up and registered a domain in G Suite (formerly known as Google Apps for Work). OpenID access to NetSuite is available from this registered domain only, not from a @gmail.com account.

2. An account administrator, or another user with the Setup OpenID Single Sign-on permission, must complete the following configuration tasks in NetSuite:
   - Enable the OpenID Single Sign-on feature.
   - Record the Google Apps domain name in NetSuite.
   - Assign the OpenID Single Sign-on permission to users who need direct access from G Suite to NetSuite.

   For steps to complete these tasks, see Setting Up Google Account Access to NetSuite.

3. To directly access NetSuite, users must enter a URL to link directly to the site, for example:
   https://system.netsuite.com/app/common/integration/openidlogin.nl?
   partner=Google&domain=example.com

The first time users connect to NetSuite from G Suite, they are required to enter their NetSuite user name and password to create a mapping between their Google identity and NetSuite identity. After this initial mapping is completed, login to NetSuite usually occurs automatically whenever the user accesses the OpenID Single Sign-on URL.

**Important:** After the initial mapping is completed, there are certain situations when users are presented with the Choose Roles page, instead of automatic login to NetSuite. These situations include when no default role has been selected and when the user has more than one role with the OpenID Single Sign-on permission.

For more information about the login process, see Logging in to NetSuite from G Suite.
Setting Up Google Account Access to NetSuite

**Note:** As an alternative to OpenID, SAML 2.0 is available for single sign-on access to NetSuite. G Suite supports SAML 2.0. For more information, see SAML Single Sign-on. Also, the G Suite Administrator Help website provides documentation on setup and configuration of SAML SSO to NetSuite. Click the following to view Google’s documentation: https://support.google.com/a/answer/6194968?hl=en. After the setup and configuration is completed, your users can use their G Suite credentials to access NetSuite.

Your organization must have previously set up and registered a domain in G Suite (formerly known as Google Apps for Work). The registered domain is required to use the OpenID Single Sign-on feature.

To allow users to access NetSuite directly from G Suite, a NetSuite account administrator needs to complete the following NetSuite configuration tasks:

1. Enable the OpenID Single Sign-on Feature
2. Record the Registered Domain Name in NetSuite
3. Assign the OpenID Single Sign-on Permission to Users

## Enable the OpenID Single Sign-on Feature

**Warning:** By enabling the OpenID Single Sign-on feature, you allow users to access and use your NetSuite account directly from a third party service. This service may not have the same authentication and security features as NetSuite. You need to ensure that NetSuite use through OpenID Single Sign-on meets all of your security, regulatory, and other compliance obligations, including Payment Card Industry (PCI) Data Security Standards.

To enable the OpenID Single Sign-on feature:

1. In NetSuite, go to Setup > Company > Enable Features.
2. On the **SuiteCloud** subtab, check the **OpenID Single Sign-on** box.
3. Review the **Terms of Service** and click **I Agree**.
4. Click **Save**.

After you save the Enable Features page, you need to go to Setup > Integration > OpenID Single Sign-on to record your Google Apps domain name.

## Record the Registered Domain Name in NetSuite

Your organization must have previously set up and registered a domain in G Suite (formerly known as Google Apps for Work). You must record the domain name on the Google OpenID Setup page in NetSuite. This page is only available after the OpenID Single Sign-on feature has been enabled in your account.
To record the registered domain name in NetSuite:

1. Go to the Google OpenID Setup page at Setup > Integration > OpenID Single Sign-on.
2. In the Google Apps Domain field, enter your organization's domain name, and click Submit.
   - The domain name should be in the format `<second level and lower level domains>.<top level domain>`. For example: `mydomain.com`.
   - Do not include parts that designate a particular host server like “www.”
   - A Google Apps domain can only be used by one NetSuite account. An error occurs if you enter a domain name already in use by another NetSuite account. This error may occur in the following cases:
     - You have incorrectly typed your domain name. Review the name and correct as necessary.
     - Your organization has multiple NetSuite accounts, and has recorded the Google Apps domain for another of your accounts in error.
       - You need to modify that other account by either: recording a new domain for it, or disabling the OpenID Single Sign-on feature and clearing the domain for it. Then you can enter the domain for the correct account.
     - The domain name is in use by an account owned by another organization.
       - If you receive an error that the domain is already in use, and you have not recorded it for one of your accounts, contact NetSuite Customer Support for assistance.

After you have submitted the Google OpenID Setup page, the Manage Roles list page displays, so you can assign the OpenID Single Sign-on permission as necessary.

Assign the OpenID Single Sign-on Permission to Users

You can provide Google OpenID access to NetSuite users by giving them the OpenID Single Sign-on permission. Generally, NetSuite permissions are assigned on a per-role basis. Each user may have multiple roles. To access NetSuite from G Suite (formerly known as Google Apps for Work), a user must have at least one role with this permission.

If you want to ensure that a user accesses NetSuite ONLY through OpenID single sign-on, you can assign that user a single role that is designed as Single Sign-on Only. For details, see the help topic Setting Up a Single Sign-on Only Role.
To add the OpenID Single Sign-on permission to a role:

1. Go to Setup > Users/Roles > Manage Roles.
2. Click the Edit link or Customize link next to the role where you want to add the permission.
3. On the Role page, Permissions subtab, Setup subtab, select OpenID Single Sign-on and click Add.
4. Click Save.
5. Repeat these steps for all roles to which you want to add this permission.

Note the following:

- The OpenID Single Sign-on permission is not provided by default to the Administrator role and cannot be added to this role.
- Be aware that permissions can only be added to customized roles. To add a permission to a standard role, you need to create a customized version of it. For more information about NetSuite roles, see the help topic NetSuite Roles Overview.
- If the Global Permissions feature is enabled in your account, you can assign the Google OpenID permission on a per-user basis, to apply across all of a user's roles. For information about this feature, see the help topic Using the Global Permissions Feature.

OpenID Single Sign-on Access for Center Roles

You can add the OpenID Single Sign-on permission to customized versions of the following center roles: Employee Center, Vendor Center, Partner Center, Advanced Partner Center, NetSuite Support Center, and NetSuite Support Center (Basic). Center roles are different from other NetSuite roles in that you can only add a limited set of permissions to them.

To add the OpenID Single Sign-on permission to a customized center role:

1. Go to Setup > Users/Roles > Manage Roles.
2. Click the Edit link for a customized center role or click the Customize link for a standard center role.
3. On the Role page, go to the Permissions subtab.
4. On the Setup subtab, set the Level to Full for the OpenID Single Sign-on permission.

Assigning the Set Up OpenID Single Sign-on Permission

When the OpenID Single Sign-on feature is enabled, in addition to the OpenID Single Sign-on permission, the Set Up OpenID Single Sign-on permission is available to be assigned to NetSuite roles (not including center roles). This permission allows users other than NetSuite account administrators to view and edit the Google OpenID Setup page.
Setting Up Google Account Access to NetSuite

Note: Administrators already have this permission.

Logging in to NetSuite from G Suite

Note: As an alternative to OpenID, SAML 2.0 is available for single sign-on access to NetSuite. G Suite supports SAML 2.0. For more information, see SAML Single Sign-on. Also, the G Suite Administrator Help website provides documentation on setup and configuration of SAML SSO to NetSuite. Click the following to view Google's documentation: https://support.google.com/a/answer/6194968?hl=en. After the setup and configuration is completed, your users can use their G Suite credentials to access NetSuite.

After an account administrator has completed the steps for Setting Up Google Account Access to NetSuite, users can log in to G Suite, and then access NetSuite.

Important: To directly access NetSuite, users must enter a URL to link directly to the site, for example:

https://system.netsuite.com/app/common/integration/openidlogin.nl?partner=Google&domain=example.com

The first time a user accesses the OpenID Single Sign-on URL in a browser, a NetSuite login page displays, where the user needs to enter a NetSuite user name and password, to create a mapping between their Google identity and NetSuite identity.

After this information is entered:

- If the OpenID Single Sign-on feature has not been enabled or a Google Apps domain name has not been recorded for the NetSuite account, an error occurs.
- If a user does not have any roles with the OpenID Single Sign-on permission, an error occurs.
- If a user has one role with this permission, the user is logged in with this role and taken to their NetSuite Home page.
- If a user has multiple roles with this permission, a Choose OpenID Role page is displayed, where the user can choose a role to use to log in, and optionally set a default role for OpenID access to NetSuite. The user is logged in with the chosen role and taken to their NetSuite home page.

Note: A user’s default OpenID role can be different from their general NetSuite default role.

When a user logs into NetSuite the first time, the initial mapping is completed between NetSuite and Google Account. After the initial mapping is completed, login to NetSuite usually occurs automatically whenever the user accesses the OpenID Single Sign-on URL.
Logging in to NetSuite from G Suite

**Important:** After the initial mapping is completed, there are certain situations when users are presented with the Choose Roles page, instead of automatic login to NetSuite. These situations include: when no default role has been selected and when the user has more than one role with the OpenID Single Sign-on permission.

A user that has multiple roles with OpenID Single Sign-on permission can change among these roles during an OpenID session. If a user attempts to change to a role that does not have the OpenID Single Sign-on permission, a login page displays for the user to log in to a non-OpenID NetSuite session.

If a timeout occurs due to lack of use during a NetSuite OpenID session, the user is logged out, instead of the user interface being locked, as occurs in other NetSuite sessions.

Removing Google OpenID Access to NetSuite

**Important:** As an alternative to OpenID, SAML 2.0 is available for single sign-on access to NetSuite. G Suite supports SAML 2.0. For more information, see SAML Single Sign-on. Also, the G Suite Administrator Help website provides documentation on setup and configuration of SAML SSO to NetSuite. Click the following to view Google's documentation: https://support.google.com/a/answer/6194968?hl=en. After the setup and configuration is completed, your users can use their G Suite credentials to access NetSuite.

There are multiple ways to remove Google OpenID access to NetSuite.

- Either of the following actions removes Google OpenID access to NetSuite for a user or group of users in your account:
  - Removing the OpenID Single Sign-on permission from the users' roles.
  - Editing the users' employee records in NetSuite to make them inactive.

- The following action removes Google OpenID access to NetSuite for all users in your account:
  - Disabling the OpenID Single Sign-on feature in NetSuite. (You can leave the domain name populated to preserve mappings, if you intend to enable the feature again in the future.)
Digital Signing

Digital signing provides authentication of documents or messages so that the identity of the sender and the validity of the document's contents can be trusted. Some organizations require digital signing of electronic documents, such as invoices, using official digital certificates. NetSuite can store digital certificates that your company has acquired, and you can use these certificates to digitally sign your documents. NetSuite stores these certificates securely, tracks the expiration dates of the certificates, and reminds users with appropriate role access when a certificate's expiry date is approaching. Users in NetSuite OneWorld accounts can upload digital certificates for multiple subsidiaries.

Note: You do not need to provide information for public certificates. NetSuite manages public certificates for key pairing.

When digital certificates are stored in NetSuite, developers can create customizations to digitally sign transactions, documents, or reports in XML or plain string format using the SuiteScript 2.0 API. For example, with SuiteScript, you can create a search for transactions that need to be digitally signed with your company certificate before sending to the customer or vendor. Your script can iterate through the search results, convert each transaction to XML, add an encrypted digital signature to a portion of the XML, and send the transaction to its recipient.

Your private digital certificates are not stored in the File Cabinet but can be uploaded on the Digital Certificates page at Setup > Company > Preferences > Certificates. Other than Administrators, only users with custom roles that include specific permissions for certificate access can upload or access private certificate information. For more information, see Uploading Digital Certificates and Access to Digital Certificates.

The Digital Signing API consists of three SuiteScript 2.0 modules:

- N/https/clientCertificate Module
- N/crypto/certificate Module
- N/certificateControl Module

Uploading Digital Certificates

You can store and manage your digital certificates on the Digital Certificates page at Setup > Company > Preferences > Certificates. The following certificate file types are currently accepted:

- PFX
- P12
- PEM

Note: Regardless of user role, you cannot download digital certificates.

To upload a new certificate:

1. Go to Setup > Company > Preferences > Certificates.
2. At the top of the page, click the Create New button.
3. In the New Certificate window, on the Details tab, enter a descriptive name for this certificate in the Name field.
4. In the ID field, enter the script ID for this certificate. The script ID of the certificate and lets you access the certificate using SuiteScript. You should make this a descriptive ID with no spaces or special characters. NetSuite prefixes the script ID with 'custcertificate'.
5. In the **Description** field, enter a description of this certificate, such as its use and who maintains it.

6. On the **Files** tab, in the **Certificate File** field, choose a file to upload the digital certificate. A file type of PFX, PEM, or P12 is required in order to save this certificate.

7. In the **Password** field, enter the password for this certificate. A password is provided by the certificate authority that issued you the certificate.

8. On the **Audience** tab, check the Restrict to Employees box to limit access to this certificate to specific employees. Select the employees in the field below. You do not need to use CTRL or CMD. Click each name to select multiple employees. Employees must also have correct role access to access this certificate. See **Access to Digital Certificates**.

9. In the **Subsidiaries** field, select which subsidiaries this certificate applies to. You can select more than one or check the box at the top of the list to select all subsidiaries. Selecting a subsidiary allows you to search for certificates by subsidiary and does not affect access.

10. Under **Expiration Reminders**, select how far in advance of the expiration date you would like Administrators to receive a reminder: one week, one month, or three months. You can select more than one option to receive more than one reminder.

11. Check the **Copy Employees** box to copy additional employees on reminders. Select which employees to copy in the field below. You do not need to use CTRL or CMD. Click each name to select multiple employees.

12. Click **Save**. The certificate file is decrypted and validated using the provided password. The certificate and password are securely stored to the NetSuite database.

**Note:** When testing in various environments, you must re-upload your certificate to the new environment. For example, if you upload a certificate in your production account and refresh your sandbox account, you must still re-upload your certificate in the sandbox account.

You can view the list of uploaded certificates on the Digital Certificates page.

### Access to Digital Certificates

If you are not using the Administrator role, you need a custom role with the Certificate Management permission in order to view the Digital Certificates page and upload new certificates.

The following role permissions apply to digital certificates and the Digital Signing API:

- **Certificate Management** — This permission controls access to the Digital Certificates page in the UI
- **Certificate Access** — This permission controls access through scripting. When you select a custom role with this permission in the Execute As Role field on script deployments, the script can access the digital certificate data for digital signing.

**Note:** If a certificate record has the Restrict to Employees box checked, only the selected employees have access to that certificate. Selected employees must also have one of the role permissions listed.
Keys

Use SSH keys to establish an SFTP connection. By using the keys, you can manage files and directories by using the SSH file transfer (SFTP) protocol. For more information, see the help topic N/sftp Module.

NetSuite stores the keys securely. Your private keys can be uploaded on the Keys page at Setup > Company > Preferences > Keys. Other than Administrators, only users with custom roles that include specific permissions for keys access can upload or access keys information. For more information, see the help topic N/keyControl Module.

The following algorithms are supported:

- RSA
- DSA
- ECDSA

Uploading Private Keys

You can store and manage your keys on the Keys page at Setup > Company > Preferences > Keys. Keys with or without passphrase are accepted.

To upload a new key:

1. Go to Setup > Company > Preferences > Keys.
2. At the top of the page, click the Create New button.
3. In the New Private Key window, on the Details tab, enter a descriptive name for this key in the Name field.
4. In the ID field, enter the script ID for this key. The script ID of the key lets you access the key using SuiteScript. You should make this a descriptive ID with no spaces or special characters. NetSuite prefixes the script ID with ‘custkey’.
5. In the Description field, enter a description of this key, such as its use and who maintains it.
6. On the Files tab, in the Private Key File field, choose a file in OpenSSH format to upload the key. Examples of key files are id_rsa, id_ecdsa, and id_dsa.
7. In the Password field, enter the same password that you used while generating the key by using the ssh-keygen command.
8. Click Save. The key is decrypted and validated using the provided password. The key and password are securely stored to the NetSuite database.
9. In the Audience tab, select the checkbox if you want to restrict the usage of the key to the specified list of employees.

Note: When testing in various environments, you must re-upload your key to the new environment. For example, if you upload a key in your production account and refresh your sandbox account, you must still re-upload your key in the sandbox account.

You can view the list of uploaded keys on the Keys page.

Access to Keys

If you are not using the Administrator role, you need a custom role with the Key Management permission in order to view the Keys page and upload new keys.
The following role permissions apply to Keys:

- **Key Management** — This permission controls access to the Keys page in the UI.
- **Key Access** — This permission controls access through scripting. When you select a custom role with this permission in the Execute As Role field on script deployments, the script can access the keys.
RESTlet Authentication

RESTlets require authentication and calls are processed synchronously. The way to provide login credentials for a RESTlet varies according to whether the RESTlet is called from an external client or from a client hosted by NetSuite, such as a client SuiteScript.

See the following sections for information about authentication for RESTlets:

- Authentication for RESTlets
- Setting up Token-based Authentication for a RESTlet integration
- Using User Credentials for RESTlet Authentication

**Note:** RESTlets are part of SuiteScript. They are not part of NetSuite's web services feature. Be aware that if a role has the Web Services Only option set to true, a user logged in through that role is permitted to send web services calls only. RESTlet calls receive an INVALID_LOGIN_CREDENTIALS error response.

Authentication for RESTlets

RESTlets must use rest URLs to connect to NetSuite. If the RESTlet call comes from an external client, the URL must include a domain specific to your NetSuite account. This domain can change without notice. For that reason, you must dynamically discover the correct domain when calling RESTlets from an external client. To handle this task, use the roles service, as described in The REST roles Service.

- For a RESTlet called from an external client, you can use OAuth or the NetSuite-specific method NLAuth in the HTTP Authorization header. OAuth uses token-based authentication (TBA) to access resources on behalf of a user, eliminating the need to share login credentials such as username and password. See Required Data for Using TBA with RESTlets.
  
  NLAuth passes in NetSuite login credentials such as company ID, user name, password, role, and application ID. See Using User Credentials for RESTlet Authentication.

- For a RESTlet called from a client hosted by the same NetSuite account that hosts the RESTlet, you do not need to pass authentication information in the HTTP request. A check for all valid NetSuite session cookies occurs, and this existing session is reused.

**Important:** RESTlet authentication can use either the HTTP Authorization header or all session cookies, but not both. Please ensure that your script uses only one form of authentication.

Setting up Token-based Authentication for a RESTlet integration

NetSuite supports token-based authentication (TBA) a robust, industry standard-based mechanism that increases the overall security of the system. This authentication mechanism enables client applications to use a token to access NetSuite through APIs, eliminating the need for RESTlets to store user credentials. A token is valid for one specific company, user entity, and role only.
Setting up Token-based Authentication for a RESTlet integration

**Important:** All encoding in TBA is percent encoding. Strings must be escaped using RFC 3986. If you do not escape characters in the header, you may receive an INVALID_LOGIN_ATTEMPT error. For more information about percent encoding, go to [https://tools.ietf.org/html/rfc5849#section-3.6](https://tools.ietf.org/html/rfc5849#section-3.6).

**Note:** Web Services Only roles are only for access to NetSuite through web services. Roles with the Web Services Only restriction will not work with RESTlets.

For more information, see [Getting Started with Token-based Authentication](https://tools.ietf.org/html/rfc5849#section-3.6).

When you use token-based authentication, password rotation policies in the account do not apply to tokens and password management is unnecessary for your RESTlets integrations. Token-based authentication allows integrations to comply with any authentication policy that is deployed in a NetSuite account for UI login, such as SAML Single Sign-on, Inbound Single Sign-on, or Two-Factor Authentication. To enable token-based authentication, see [Enabling the Token-based Authentication Feature](https://tools.ietf.org/html/rfc5849#section-3.6).

You can create a token and assign it to a user by logging in to NetSuite as an administrator and generating token credentials manually. NetSuite users can also generate token for themselves. See [Token-based Authentication (TBA) Permissions](https://tools.ietf.org/html/rfc5849#section-3.6).

For code snippets and examples of signature creation and token-based authentication, see [SuiteAnswer 42171](https://tools.ietf.org/html/rfc5849#section-3.6) and [SuiteAnswer 42019](https://tools.ietf.org/html/rfc5849#section-3.6).

For information about calling a token endpoint to issue or revoke a token, see the help topic [Issue Token and Revoke Token REST Services for Token-Based Authentication](https://tools.ietf.org/html/rfc5849#section-3.6) in the Token-based Authentication section of the Help Center.

**Using TBA for RESTlet Authentication (OAuth)**

If appropriate, you can use NetSuite’s Token-Based Authentication feature to authenticate when calling RESTlets. With this approach, you use the OAuth 1.0 specification to construct an authorization header. For details, see the following topics:

- **TBA Setup Requirements**
- **Required Data for Using TBA with RESTlets**
- **Example OAuth Header**
- **Tracking RESTlet Calls Made with TBA**

**Note:** If you are calling a RESTlet from an external source, you must authenticate by using either TBA or user credentials. For details on user credentials, see [Using User Credentials for RESTlet Authentication](https://tools.ietf.org/html/rfc5849#section-3.6).

**TBA Setup Requirements**

Using the OAuth protocol with RESTlets requires NetSuite’s Token-based Authentication (TBA) feature. Before you can use TBA, you must complete several setup tasks. These tasks include the following:

- You must have enabled the Token-based Authentication feature. For details, see [Enabling the Token-based Authentication Feature](https://tools.ietf.org/html/rfc5849#section-3.6).
- You must have created a role that permits logging in using token-based authentication. For details, see [Set Up Token-based Authentication Roles](https://tools.ietf.org/html/rfc5849#section-3.6).
- You must have assigned a user to a role that has permission to log in by using token-based authentication. For details, see [Assigning Users to Token-based Authentication Roles](https://tools.ietf.org/html/rfc5849#section-3.6).
An integration record representing the sending application must exist at Setup > Integration > Manage Integrations. On the integration record, the Token-based Authentication option must be enabled. Enabling this option causes the system to generate the consumer key and secret that represent the application. For details, see Creating an Integration Record.

You must have the consumer key and secret that were generated when the integration record’s Token-based Authentication option was enabled. If you do not have these credentials, you can generate new ones. For details, see Regenerating a Consumer Key and Secret.

You must have created a token and token secret for the user who will call the RESTlet. For details on this process, see Managing TBA Tokens in the NetSuite UI.

**Note:** For general details about NetSuite's Token-based Authentication feature, see Token-based Authentication (TBA).

After you have verified that the prerequisite steps have been completed, you can create logic for generating an OAuth header. For details on the data required for creating the header, see Required Data for Using TBA with RESTlets.

### Required Data for Using TBA with RESTlets

An OAuth 1.0 RESTlet authorization header requires the data described in the following table. Some of these values can be obtained from the NetSuite UI. Other values must be calculated. Typically, your integration should include logic to identify these values and generate the finished header. Follow the OAuth 1.0 protocol to create the authorization header.

**Important:** All encoding in TBA is percent encoding. Strings must be escaped using RFC 3986. If you do not escape characters in the header, you may receive an INVALID_LOGIN_ATTEMPT error. For more information about percent encoding, go to [https://tools.ietf.org/html/rfc5849#section-3.6](https://tools.ietf.org/html/rfc5849#section-3.6).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>realm</td>
<td>The ID of the NetSuite account where the RESTlet is deployed.</td>
<td>You can find this value at Setup &gt; Integration &gt; Integration Management &gt; SOAP Web Services Preferences, in the Account ID field.</td>
</tr>
<tr>
<td>oauth_consumer_key</td>
<td>The consumer key for the integration record being used to track the calling application. This string was created when you checked the Token-based Authentication box on the integration record and saved it. To create an OAuth header, you also need the consumer secret that goes with the key, although you do not use the secret explicitly. If you no longer have these values, you can regenerate them. For details, see Regenerating a Consumer Key and Secret.</td>
<td></td>
</tr>
<tr>
<td>oauth_token</td>
<td>A token that represents a unique combination of a user and an integration record. This string can be generated in multiple ways. For details, see Managing TBA Tokens in the NetSuite UI. To create an OAuth header, you also need the token secret that goes with the token, although you do not use the secret explicitly.</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>oauth_nonce</td>
<td>A unique, randomly generated alphanumeric string of 6-64 characters.</td>
<td></td>
</tr>
<tr>
<td>oauth_timestamp</td>
<td>A current timestamp in Unix format.</td>
<td></td>
</tr>
<tr>
<td>oauth_signature_method</td>
<td>A hash algorithm that can be used to create an RFC 2104-compliant signature.</td>
<td>Supported choices are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ HMAC-SHA1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ HMAC-SHA256</td>
</tr>
<tr>
<td>oauth_version</td>
<td>The version of OAuth being used.</td>
<td>Only one value is supported: 1.0.</td>
</tr>
<tr>
<td>oauth_signature</td>
<td>A signature generated as described in Section 3.4 of RFC 5849, which describes the OAuth 1.0 specification: <a href="https://tools.ietf.org/html/rfc5849">https://tools.ietf.org/html/rfc5849</a>.</td>
<td>To create the signature, you need all of the other values listed in this table, among others, such as the HTTP method being used to make the call. With many languages, an OAuth library is available to help you create the signature. For details about some of the third-party open source libraries that are available, see SuiteAnswer 42171. If you are working in a language that does not have a library, you may want to refer to SuiteAnswer 42019 for an overview of the signature-creation process.</td>
</tr>
</tbody>
</table>

For an example of the finished header, see Example OAuth Header.

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

### Example OAuth Header

The following snippet shows a correctly formatted OAuth header.

**Important:** All encoding in TBA is percent encoding. Strings must be escaped using RFC 3986. If you do not escape characters in the header, you may receive an INVALID_LOGIN_ATTEMPT error. For more information about percent encoding, go to https://tools.ietf.org/html/rfc5849#section-3.6.

```
Authorization:
    OAuth realm="12345",
    oauth_consumer_key="4a3ff6c251a55057bb1e62d8dc8999a036e688f3a8fe735265fc425368b0f154",
    oauth_token="52cfe88f82e2b74e833e7dfc4c4ae79ff44c3ca9f696d61e2a7eeac6c8357c3c",
    oauth_nonce="qUwImPtGCS4sHJe8F7x",
    oauth_timestamp="1462453273",
    oauth_signature_method="HMAC-SHA1", oauth_version="1.0",
```
Setting up Token-based Authentication for a RESTlet integration

For more information, log in to SuiteAnswers and review the following articles.

⚠️ Important: For help logging in to SuiteAnswers, see the help topic SuiteAnswers Overview. You must log in to SuiteAnswers before you can access the following links.

- OAuth Library Consumption for Client Application
- C# > RESTlet Authentication Using Token (Token-Based Authentication)
- Python via cURL > RESTlet Authentication using Token (Token-Based Authentication)
- Java > RESTlet Authentication using Token (Token-Based Authentication)
- Suitelet > RESTlet Authentication using Token (Token-Based Authentication)
- Python > POST using Token-based Authentication
- SuiteScript > POST using Token-based Authentication
- Error Message "Invalid Login Attempt" on RESTlet when authenticating using Tokens

Tracking RESTlet Calls Made with TBA

If you use OAuth headers when calling RESTlets, you have the ability to track and block RESTlet calls. You manage RESTlet activity by using integration records. Each record shows the RESTlet calls that authenticated by using that record's consumer key.

Integration records are located at Setup > Integration > Manage Integrations. For more information on using integration records in conjunction with RESTlets, see the following topics:

- Creating an Integration Record
- Blocking an Application
- Enabling an Application to Use Token-based Authentication
- Regenerating a Consumer Key and Secret
- Using the RESTlets Execution Log
- Ownership of Integration Records
- Tracking Changes to Integration Records

Note: For more information about managing integration records, see the help topic Integration Management, which is part of the SOAP Web Services Platform Guide. However, be aware that some of the detail in that guide pertain only or primarily to SOAP web services.

Creating an Integration Record

To create an OAuth header, you must have a consumer key and secret that represents the application that will call the RESTlet. In general, you create these values by creating an integration record. When you create the record and enable the record's Token-based Authentication option, the system generates and displays the consumer key and secret. These values are shown only one time. However, if you lose the values, you can regenerate them, as described in Regenerating a Consumer Key and Secret.

In some cases, you might be working with a partner that has provided an integration record to you through a bundle. In this situation, you obtain the consumer key and secret from the partner. If you have an integration record that was installed through a bundle, note that not all fields on the record
are modifiable. Non-modifiable fields include Name, Description, User Credentials, and Token-based Authentication.

To create an integration record, use the following procedure.

**Note:** The integration record is also used to track web services activity. Not all fields on the record are relevant to RESTlet activity. For full details on how this record is used with SOAP web services, see the help topic **Integration Management**.

**To manually create an integration record:**

1. Go to Setup > Integration > Manage Integrations > New.
2. In the **Name** field, enter a name for the application.
3. If appropriate, enter a description in the **Description** field.
4. If you want NetSuite to block requests from this application, set the **State** dropdown list to **Blocked**. Otherwise, leave this field set to its default of **Enabled**. If you intend to distribute this record, be aware that the value you choose for this field is not propagated to accounts where the record is installed. The value of this field is always specific to one NetSuite account. If the record is installed by bundling, the State field is always initially set to Enabled.
5. If appropriate, enter additional details about the application in the **Note** field. If you intend to distribute this record, be aware that the text you enter in this field is not visible in accounts where the record is installed. The value of this field is always specific to one NetSuite account.
6. On the **Authentication** subtab, select the authentication methods that this application should be permitted to use. You can choose one or both of the following:
   - **User credentials** – This option is applicable only to SOAP web services activity. For help using this option in conjunction with SOAP web services, see the help topic **Sending an Application ID with User Credentials**.
   - **Token-based authentication** – This option lets the application use token-based authentication (TBA).
7. Click **Save**.

   The system saves the new record. If you selected the **Token-based Authentication** option, the updated page also shows the record’s consumer key and secret.

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

**Blocking an Application**

If appropriate, you can block an application represented by an integration record. Blocking an application has the following effects:

- The application is blocked from authenticating using the consumer key associated with the integration record. So, if an application is using an OAuth header to call RESTlets (using data from this integration record), these calls will be blocked,
- If the application makes SOAP web services requests, the requests are blocked if they reference either the consumer key or the application ID associated with the integration record.

Note that this procedure does not prevent an application from calling a RESTlet by using the NLAuth authentication method. Similarly, the application is not blocked if it already has an existing session.
To block an application:

1. Navigate to Setup > Integration > Managing Integrations, and open the appropriate integration record for editing.
2. Set the **State** field to **Blocked**.
3. Click **Save**.

Enabling an Application to Use Token-based Authentication

In some cases, you might have an existing application that is not set up for token-based authentication. For example, an integration record might have been created to track SOAP web services activity, and that application might authenticate through user credentials. If appropriate, you can enable token-based authentication for that application.

**Note:** If the integration record was created in another account and installed in your account through a bundle, you cannot modify the Token-based Authentication field. For more details, see Ownership of Integration Records.

To enable token-based authentication for an existing application:

1. Navigate to Setup > Integration > Managing Integrations, and open the appropriate integration record for editing.
2. Check the **Token-based Authentication** box.
3. Click **Save**.

   The system displays the consumer key and secret on the screen. Make a note of these values. You will need them to create an OAuth header.

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

Regenerating a Consumer Key and Secret

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

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

**Note:** If the integration record was created in another account and installed in your account through a bundle, you cannot reset the credentials. The credentials can be reset only by an authorized user in the NetSuite account where the record was created. For more details, see Ownership of Integration Records.

To regenerate a consumer key and secret:

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

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

### Using the RESTlets Execution Log

Each integration record includes a subtab labeled RESTlets under the Execution Log tab. This log lists RESTlet calls that are uniquely identified with that integration record. That is, the log includes those requests that use token-based authentication and reference the integration record’s consumer key.

**Note:** Calls made using the NLAuth method of authentication are not logged on any integration record.

For each logged request, the RESTlets Execution Log includes details such as the following:

- The date and time that the call was made.
- The duration of the request.
- The email address of the user who made the request.
- The action taken.
- The corresponding script ID and deployment ID.

### Ownership of Integration Records

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

You can also install records in your account that were created elsewhere. For example, an integration record could be bundled and distributed. If you install a bundle that includes an integration record, the record is considered to be an installed record. It is owned by a different NetSuite account. On such records, you can make changes to only two fields: the Note field and the State field. All other fields, including the authentication and Description fields, can be changed only by an authorized user in the account that owns the record. When the owner makes changes to these fields, the new settings are pushed automatically to your account. These changes are not reflected in the system notes that appear in your account.

### Tracking Changes to Integration Records

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

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

Be aware that in general, system notes are created only for those fields that you are permitted to change. For additional details, see the help topic **Special Cases Related to System Notes Logging**. Note that some of the information in that topic is specific to SOAP web services.

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

- By using the System Note search type, at Reports > New Search.
- By clicking on the System Notes subtab of any integration record.

Using User Credentials for RESTlet Authentication

When you call a RESTlet, you can authenticate by providing a user ID and password. With this approach, you use an NLAuth authorization header. NLAuth is a NetSuite-specific authentication approach that is used for RESTlets only.

**Note:** If you are calling a RESTlet from an external source, you must authenticate by using either user credentials or token-based authentication. For details on TBA, see **Using TBA for RESTlet Authentication (OAuth)**.

**Required Data**

To construct an NLAuth authorization header, you use the fields described in the following table.
**Important:** Strings must be escaped using RFC 3986. If you do not escape characters in the header, you may receive an INVALID_LOGIN_ATTEMPT error. For more information about percent encoding, see [https://tools.ietf.org/html/rfc5849#section-3.6](https://tools.ietf.org/html/rfc5849#section-3.6).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Required?</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlauth_account</td>
<td>The ID of the NetSuite account where the RESTlet is deployed.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>nlauth_email</td>
<td>The email address with which the user logs in to NetSuite.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>nlauth_signature</td>
<td>The user’s password.</td>
<td>Yes</td>
<td>If you omit this value, the system selects a role based on the logic described in RESTlet and SOAP Web Services Role Selection Logic.</td>
</tr>
<tr>
<td>nlauth_role</td>
<td>The internal ID of a role with which the user is associated.</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>nlauth_application_id</td>
<td>The application ID of the integration associated with the RESTlet.</td>
<td>No</td>
<td>The application ID is not required, but it is highly recommended. By adding an application ID, you associate RESTlets with a specific integration. By associating your RESTlets with an integration record, you can take advantage of the benefits of integration records. The benefits include support for viewing details about your integration applications, blocking an application, and viewing the execution log specific to each application. For more information, see the help topic Integration Record Overview.</td>
</tr>
</tbody>
</table>
| nlauth_otp      | The value of the one-time password (OTP) is the same as the value of a two-factor authentication (2FA) verification code generated by an authenticator app when a user is logging in to the NetSuite UI. | No        | For the issue token endpoint, including the nlauth_otp parameter in the NLAuth authorization header permits the sending of an OTP. The OTP is a 2FA verification code. For more information, see the following topics:  
  - Mandatory 2FA, the IssueToken Endpoint, and nlauth_otp  
  - Issue Token and Revoke Token REST Services for Token-Based Authentication |

**RESTlet and SOAP Web Services Role Selection Logic**

In 2017.1, the logic for how a role is selected when no role is specified in the request changed. This change does not affect logins where a role is provided.

This type of login is possible for each of the following:

- RESTlet calls that use an NLAuth authorization header
- SOAP web services requests that use the following login approaches:
  - The login operation
  - Request-level credentials with the Passport complex type
If a Customer Center role must be used in an integration, you should explicitly specify the role. If no role is specified, the system chooses a role. The system tries to use a non-Customer Center role. If there are no available non-Customer Center roles, login is attempted with a Customer Center role. The overall order of role selection is:

1. The role specified by the request.
2. If the request is a SOAP web services request, the default web services role for the user, if one exists. Default SOAP web services roles are listed on the SOAP Web Services Preferences page (at Setup > Integration > SOAP Web Services Preferences). This role can be a Customer Center or non-Customer Center role.
3. The default role for the user, if the default role is a non-Customer Center role.
4. The last non-Customer Center role that the user logged in with.
5. The default role for the user, if the default role is a Customer Center role.
6. The last Customer Center role that the user logged in with.

For more information about specifying a role in a RESTlet or SOAP web services request, see:

- the Passport complex type
- the login operation

**Important:** RESTlet authentication accepts special characters only if they are URL encoded. If your credentials contain special characters, replace each special character with its appropriate URL encoding. For additional information on URL encoding, see http://www.w3schools.com/tags/ref_urlencode.asp.

**Syntax**

The NLAuth header requires the following elements:

- The prefix NLAuth, followed by a space.
- A series of field-value pairs. Each pair should include the field name, an equals sign, and a value. Separate the pairs by commas. You should enter the key-value pairs without leading or trailing spaces. For example, `nlauth_account=123456`, rather than `nlauth_account= 123456 ,`

**Examples**

The following snippet shows a correctly formatted NLAuth header.

```
Authorization: NLAuth nlauth_account=123456, nlauth_email=jsmith@example.com, nlauth_signature=xxxx,
nlauth_role=37, nlauth_application_id=12345ABC-123A-456B-789C-123456789ABC
```

The following snippet shows a correctly formatted NLAuth header when using the token endpoint. The header includes a verification code (an OTP for passing a 2FA challenge) using the nlauth_otp parameter.

```
Authorization: NLAuth nlauth_account=123456, nlauth_email=jsmith@example.com, nlauth_signature=xxxx,
nlauth_otp=654321
```

For an example of a shell script that generates an NLAuth header, see the help topic **Example: Shell Script that Calls a RESTlet.**