ORACLE NETSUITE

Authentication Guide

2022.2

March 15, 2023

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

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

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility.

Access to Oracle Support

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

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

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

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

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 administrators include Password Requirements and Policies in NetSuite, NetSuite Login Pages, and Enabling and Creating IP Address Rules.

Two-Factor Authentication (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
- Required 2FA, the IssueToken Endpoint, and nlauth_otp
- Two-Factor Authentication (2FA)
- Permissions Requiring Two-Factor Authentication (2FA)
- Designate Two-Factor Authentication Roles

Single Sign-on (SSO) Overview

NetSuite supports various 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 OpenID Connect (OIDC) Single Sign-on for inbound SSO from the OIDC Provider (OP) of your choice. See also OpenID Connect (OIDC) Access to Web Store.

- 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.

NetSuite supports two outbound single sign-on methods:

- See NetSuite as OIDC Provider. You should use NetSuite as OIDC Provider as an authentication method for outbound single-on.
- See Outbound Single Sign-on (SuiteSignOn). SuiteSignOn access to NetSuite from your web store is supported. For more information, see the help topic Outbound Single Sign-on (SuiteSignOn) Access from Your Web Store.

Authentication for API Access to NetSuite

NetSuite offers Token-based Authentication (TBA) and OAuth 2.0, enabling client applications to use a token to access NetSuite through APIs. TBA and OAuth 2.0 eliminate the need for RESTlets and web services integrations to store user credentials. You should not employ user credentials as an authentication method for web services integrations or for RESTlets.

Note: OAuth 2.0 cannot be used with SOAP web services. For more information, see Authentication Matrix.

For more information, see the following topics:

- Token-based Authentication (TBA)
  - The Three-Step TBA Authorization Flow
  - The IssueToken Endpoint
- OAuth 2.0
  - OAuth 2.0 Authorization Code Grant Flow
- Integration Management
- Authentication for SOAP Web Services
- Authentication and Session Management for REST Web Services
- Authentication for RESTlets
- Mandatory Two-Factor Authentication (2FA) for NetSuite Access

NetSuite supports two outbound single sign-on authentication methods for integrations:

- See NetSuite as OIDC Provider. The NetSuite as OIDC Provider feature is only supported for RESTlets and REST web services.
- See Outbound Single Sign-on (SuiteSignOn). SuiteSignOn is only supported for SOAP web services.

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 (SCIS).

Authentication Matrix

The following table shows the authentication methods supported in NetSuite.

	NetSuite Application	SuiteCommerce	SOAP web services	REST web services	SuiteScript RESTlets
User Credentials	Supported	Supported	You should not employ user credentials for SOAP web services. Use Token-based Authentication instead. Currently supported, with the exception of 2FA-required roles. **Important:** As of 20.2 endpoint, user credentials are not supported for using with SOAP web services.	—	**Important:** As of January 1, 2021, user credentials are not supported for using with RESTlets.
Token-based Authentication (TBA)	—	—	Supported. You should use TBA for SOAP web services authentication.	Supported. You should use TBA or OAuth 2.0 for REST web services authentication.	Supported. You should use TBA or OAuth 2.0 for RESTlet authentication.
OAuth 2.0	—	—	—	Supported. You should use OAuth 2.0 or TBA for REST web services authentication.	Supported. You should use OAuth 2.0 or TBA for RESTlet authentication.
Two-Factor Authentication (2FA)	Supported 2FA is required for highly privileged roles.	—	—	—	—
SAML 2.0	Supported	Supported	—	—	—
OpenID Connect (OIDC) Single Sign-on	Supported	Supported	—	—	—

Warning: The NetSuite Inbound SSO feature is deprecated. Migrate your solutions to a different single sign-on feature:

- Use the OpenID Connect (OIDC) Single Sign-on feature released with 2019.2. For more information, see OpenID Connect (OIDC) Single Sign-on.
- Another alternative is to use the SAML Single Sign-on feature for access to NetSuite. For more information, see SAML Single Sign-on.
- You can use the Token-based Authentication feature for SOAP web services integrations. For more information, see Token-based Authentication (TBA).

As of 2021.1, any solutions still using the Inbound SSO do not work.

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 **Two-Factor Authentication Required** 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).

2FA requirement also applies to all non-UI access. Non-UI access means accessing 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.

Password Requirements and Policies in NetSuite

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

- NetSuite Password Requirements
- PCI Compliance Password Requirements
- User Access Reset Tool
- Password Reset Tips for Administrators
- Password Changes Are Logged in System Notes on Entity Records

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 either the View Unencrypted Credit Cards permission or the View Unencrypted ACH Account Numbers permission. See PCI Compliance Password Requirements.

Note: Users are locked out for 30 minutes after five consecutive attempts to log in to NetSuite with an incorrect password. For more information, see User Access Reset Tool.

Password Settings That Can Be Modified

Password settings can be modified by an administrator at Setup > Company > General Preferences. See the following topics 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.

> **Note:** The Strong password policy was set as the default for each account in 2014.1. The Strong policy has been enforced for all new users added after 2014.1. However, this policy was only enforced for users who existed before the upgrade when these users changed their passwords. For information about how often users must change their passwords in your account, Password Expiration in Days.

- It is possible to reset the password policy to Medium or Weak, but changing password policy to less strict weakens the security of the account.

> **Warning:** If any users in your account have the View Unencrypted Credit Cards permission or the View Unencrypted ACH Account Numbers 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. Be aware of 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 eight 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. If you change this value, you can prompt your employees to change their passwords on their next login. You can check the **Require Password Change on Next Login** box on employee records. You can also use CSV import to update this option on many employee records at the same time.

- 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, 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 maximum of 180 days.

- To comply with Payment Card Industry (PCI) standards, employees with access to view unencrypted credit card numbers or unencrypted ACH account 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 an administrator:

- 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 `abcd123456` 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:

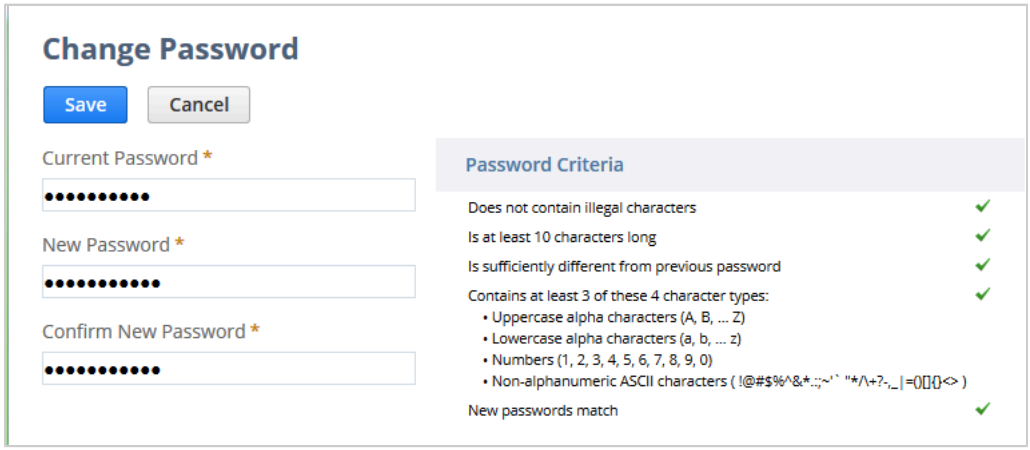
  - Uppercase alphabet (A, B, ... Z)
  - Lowercase alphabet (a, b, ... z)
  - Number (1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
  - Non-alphanumeric ASCII characters, for example ` ~ ! @ # $ % ^ & *) ; ' [ ] "{ }.

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.



PCI Compliance Password Requirements

When using features that allows viewing credit card or ACH account data, be aware of the Payment Card Industry Data Security Standard (PCI DSS) password requirements. Users with either the View Unencrypted Credit Cards permission or the View Unencrypted ACH Account Numbers permission must change their NetSuite passwords at least every 90 days.

If the number of days set in the **Password Expiration in Days** field on the General Preferences page is less than 90 days, that requirement remains in effect. For example, if your company is set to expire passwords every 60 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 either the View Unencrypted Credit Cards permission or the View Unencrypted ACH Account Numbers permission.

In addition, passwords for those with access to unencrypted credit card numbers or unencrypted ACH accounts must have a minimum of seven 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 or unencrypted ACH accounts 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 2FA phone number in NetSuite. Users should try these self-service methods before requesting help from an Administrator or a role with Core Administration Permissions (CAP).

The following topics are intended for all NetSuite users:

- Getting Access When You Forget Your Password
- Update Security Questions Link

- Reset Your 2FA Settings
- Finding Your Settings Portlet

When these self-service methods are not sufficient to resolve the problem, users need assistance. The User Access Reset page provides one place for users with an Administrator role or a role with Core Administration Permissions (CAP) to assist other users who need help with:

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

Important: To initiate a password reset for a user who has access to multiple NetSuite accounts, you must be an Administrator in all of those accounts. The User Access Reset Tool is also available to users with Core Administration Permissions, but the same restriction applies. Users who are not Administrators but have only Core Administration Permissions cannot reset the password for a user who has access to multiple NetSuite accounts. (See the help topic Core Administration Permissions if you need more information about this feature.)

To use the User Access Reset Tool:

1. In an Administrator role or a role with Core Administration Permissions (CAP), go to Setup > Users/ Roles > User Management > User Access Reset Tool.
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.
    a. **Initiate Password Reset** – check this box to send an email to the user containing a link so that the user can reset the NetSuite password.
    b. **Clear User's 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.
    c. **Unlock The User's Access** – check this box to unlock NetSuite access for a user who is locked out of NetSuite after submitting five consecutive incorrect passwords.
    d. **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 five 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 may happen, for example, when users forget the answers to their own security questions. Administrators can use the User Access Reset Tool to assist.

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.

In this section, see the following topics:

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 **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 five consecutive incorrect passwords.
  - You should use the User Access Reset Tool, but you can also 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.
- Visitors to your website can register an email address and create a password. This action creates Lead record in your NetSuite account. Lead and Prospect records can be converted into Customer records. See Automatic Reset of Customer Passwords for more information.

Automatic Reset of Customer Passwords

Not all users registering on your website remain active users, logging in again or purchasing items. These less-active users may forget the login name and password that they entered on your site. These long-abandoned passwords are automatically reset. Passwords that are automatically reset are associated with website customers who meet either of the following criteria:

- The website customer has not logged in within the previous three years.
- It has been more than 90 days since the customer registered a login name and created a password, and the customer never logged in again.
- The website customer has not logged in within 30 days after their password was changed.

The customer, lead, or prospect record is retained in your NetSuite account. Only the existing password is removed from the record. Users whose passwords have been reset may still attempt to log in. These users will receive an error message that their password has expired.

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 **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 **Forgot Your Password?**. 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

- As of 2020.1, the following requirements apply to customer passwords:
  - The minimum password length for customers is eight characters.
  - Easy to guess or potentially compromised passwords are prohibited.
- Other password policies and requirements for access to the NetSuite UI do not apply to customers:

  The value set in the account for the **Password Expiration in Days** field is not applied to customer passwords. However, customer passwords are automatically reset. See Automatic Reset of Customer Passwords for more information.

Password Changes Are Logged in System Notes on Entity Records

Note: This topic applies to System Notes only. For information about System Notes v2, see the help topic System Notes v2 Overview.

Requests to change a password are logged on the System Notes subtab of an entity record. Changes are logged no matter who or what initiates the request. System notes capture successful changes requested through the UI, web services, or RESTlets. Administrators can view the password change information in the system notes for an entity.

Changes are logged for these entity types in NetSuite: Employee, Contact, Customer, Partner, Prospect, and Vendor records. System notes include information about who or what initiated the password change, and when the change took place:

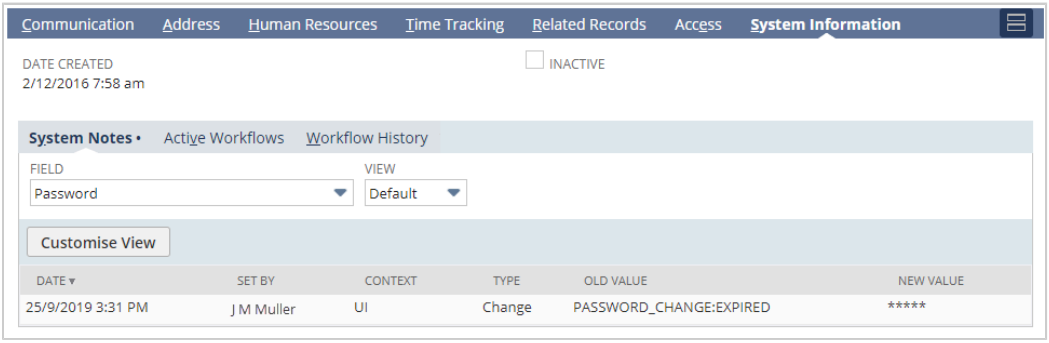
- User (from the Change Password or Forgot Your Password links, or when the user changed the password after it expired.)
- Administrators (by manual assignment to set user passwords, or by sending the **New User Access Notification Email** to let the user set the password. The Administrator can also initiate a password reset with the **User Access Reset Tool**).
- NetSuite Customer Support (using internal tools).

- Automated processes.

For more information about system notes, see the help topic System Notes Overview.

To view the system notes on an entity record:

1. Find the entity record:
   a. Go to Lists > Employees > Employees and choose the employee from the list.
   b. Go to Lists > Relationships and select the entity type (for example, Contacts, Customer, Partner, Prospect, or Vendor) as appropriate. Choose the entity from the list.
2. Click **View**.
3. Click the **System Information** subtab to view the **System Notes** subtab.
4. In the **Field** list, select Password to view only password-related system notes.



The following table describes the values for the entries in the **Context** and **Old Value** columns.

Context	Description of Values Appended to PASSWORD_CHANGE
UI	■ USER_CHANGE – The user changed the password by clicking **Change Password** link on the Settings portlet. ■ USER_RESET - The user reset the password by clicking **Forgot your password?** link on the NetSuite login page. ■ EXPIRED - The user changed the password after their old password had expired. ■ ENTITY_RECORD - The Administrator or a user with permission to modify the entity record updated the password. ■ ADMIN_RESET - An Administrator initiated a password reset from the User Access Reset Tool by checking the **Initiate Password Reset** box. ■ NEW_ACCESS - The entity was newly given access to a NetSuite account by an Administrator by one of the following methods: □ Initially setting the password manually on the entity record. □ Checking the **Send New Access Notification Email** box on the entity record.
Script	■ SUITE_SCRIPT - The password was changed programmatically through the SuiteScript API through the execution of a SuiteScript.
Web Services	■ WEB_SERVICES - The password was changed programmatically through the SuiteTalk API by a SOAP web services call.
Other	■ GENERATED - The system generated a random password for the entity. ■ TS_SET - NetSuite Customer Support set the password. ■ TS_RESET - NetSuite Customer Support reset the password.

Context	Description of Values Appended to PASSWORD_CHANGE
	■ NEW_ADMIN - In rare cases, an account has no active users with an Administrator role. NetSuite Customer Support can add the Administrator role to a user. ■ SYSTEM_TASK - A NetSuite system task set reset the password. ■ PROVISIONING - The password was set when the account was provisioned.

Automatic Reset of Long-Abandoned Passwords for Website Customers

Visitors to your website can register an email address and create a password. This action creates lead record in your NetSuite account. Lead and prospect records can be converted into Customer records. Not all users registering on your website remain active users, logging in again or purchasing items. These less-active users may forget the login name and password that they entered there.

Note: As of 2018.2, long-abandoned passwords are automatically reset. Passwords that are automatically reset are associated with website customers who meet either of the following criteria:

- The website customer has not logged in within the previous three years.
- It has been more than 90 days since the customer registered a login name and created a password, and the customer never logged in again.
- The website customer has not logged in within 30 days after their password was changed.

The customer, lead, or prospect record is retained in your NetSuite account. Only the existing password is removed from the record. Users whose passwords have been reset can still attempt to log in. These users will receive an error message that their password has expired.

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 various 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 various types of NetSuite sessions. Each type of session is managed independently from the others.

Type of Session	Timeout Values	Notes
User Interface (UI)	■ Idle session timeout: default is 180 minutes ■ Absolute session timeout: 12 hours	See User Interface (UI) Sessions for more information about UI session management and timeout values.
SOAP web services	■ Idle session timeout: 20 minutes ■ Absolute session timeout: 60 minutes ■ Operation session timeout: 15 minutes.	■ 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. ■ 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. ■ If an operation takes more than 15 minutes to complete, consider using asynchronous calls to complete the operation.
SuiteAnalytics Connect	■ Idle session timeout: 90 minutes	After 90 minutes of inactivity, SuiteAnalytics Connect sessions automatically time out. For more information, see the help topic Connections.
Web Stores (hosted by NetSuite)	■ Idle session timeout: 20 minutes	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. However, settings in the NetSuite account (like absolute and idle session timeout) may affect the website timeout value. For example, if the absolute session timeout value is set to 15 minutes, the website session will end after 15 minutes. For enhanced security, as of 2020.1, Commerce websites are subject to explicit session invalidation. Explicit session invalidation applies to all SuiteCommerce Advanced, SuiteCommerce, and Site Builder websites.

Type of Session	Timeout Values	Notes
		See the help topic Web Store Sessions for more information.

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 various 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 Offline 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 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.

Watch this video to see how you can have two or more NetSuite account types open at the same time:

Running Simultaneous NetSuite Sessions

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

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

1. Open your browser and log in to your NetSuite production account.

2. From the Change Roles list, select a sandbox account, right-click, and select open in a new tab. (The wording varies slightly depending on the browser you are using.)

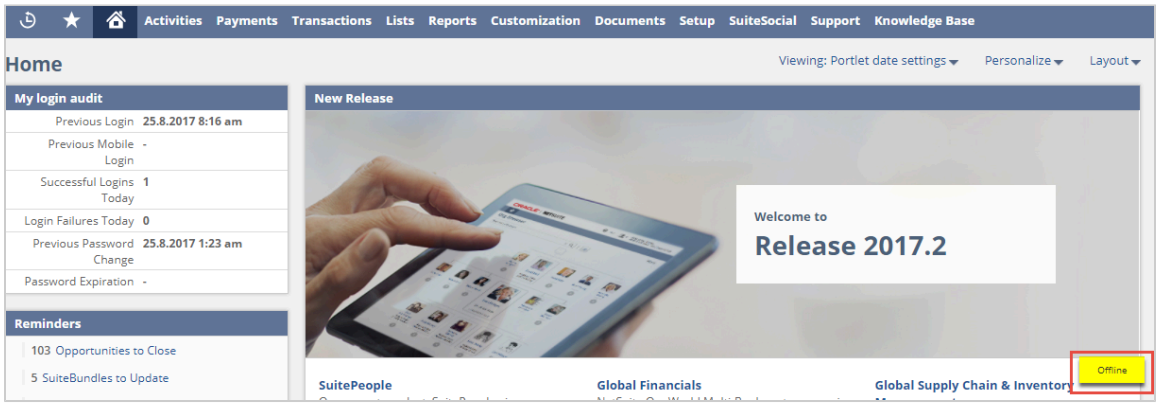
   The selected sandbox account opens in a new tab. If you do not have a default role indicated in the sandbox account, your browser history is used to automatically log you in to your last used role. For more information, see the help topic Roles and Accounts.

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 Offline 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 may 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 NetSuite 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 NetSuite 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

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

- **Standard login pages:**
  - `https://system.netsuite.com/pages/customerlogin.jsp`
  - `https://system.netsuite.com/pages/login.jsp`
  - `http://<accountID>.app.netsuite.com/app/login/secure/enterpriselogin.nl`
  - `https://system.netsuite.com/app/login/secure/enterpriselogin.nl?c=<accountID>&whence=`

  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.netsuite.com/app/login/secure/privatelogin.nl?c=<accountID>`

  Administrators can go to Setup > Company > Setup Taks > Company Information to view the **Account ID** field and **Customer Center Login** field, located in the Company URLs subtab. 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: As of 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 following code 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 following code 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:

       ```
       <form method="post" action="/app/login/secure/privatelogin.nl">
       ```

     - Modify the href line for the Forgot your password link as shown:

       ```
       <href="/app/login/preparepwdreset.nl?private=t">
       ```

   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.

```
<!--The follwing post action link is for a non-Customer Center login page-->
<form method="post" action="app/login/secure/enterpriselogin.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>
      <tr>
         <td>
            <!--The following href link is for a non-Customer Center login page-->
            <a href="/app/login/preparepwdreset.nl">Forgot your password?</a>
            <!--For Customer Center login page, modify the href link as specified in step 1.-->
         </td>
      </tr>
      <tr>
         <td>
            <input type="submit" name="submitter" value="Login" >
         </td>
      </tr>
   </table>
</form>
```

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 checked.
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 checked 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://<accountID>.app...`. 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://<accountID>.app.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.

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:

```
<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.

Using a Role Parameter

You can choose a preferred role for users to login with to the NetSuite UI. The value of the `value` parameter is the role ID.

Note: To find the role ID, go to Setup > Users/Roles > User Management > Manage Roles, and click the role name link. The role ID is visible in the role page URL. See the following example:

`https://xyz.app.netsuite.com/app/setup/role.nl?id=1047`

To choose a role for user to log in with:

1. Add a role redirect hidden field to the login form in your hosted HTML page, for example:

```
<input type="hidden" name="role" value="3" >
```

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:

```
<input type="hidden" name="errorredirect" value="/core/media/media.nl?id=572&c=TSTDRV1154923&h=b0c2553e7af5afb07ef2&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 and select the appropriate center.
3. Enter the URL to the **Log Out Landing Page** field.

Choose Role Page

You may notice an alternative version of the Choose Role page when logging in to NetSuite. This version of the page appears when the system is not able to determine your usual role or a specific account. The page appears in following cases:

- You logged in from the system domain for the first time, and there is no browser history.
- You logged in to a new account for the first time. For example, you gained access to a new production account.

Note: This version of the Choose Role page is not an error page. This page lets you explicitly choose a role or an account. The page appears because the system was not able to determine your role or account from previous NetSuite usage.

NetSuite Login Pages and iFrame Prohibition

Security policies and contractual agreements prohibit displaying a NetSuite login page in an iFrame. For more information about this prohibition, see the help topic Secure Login Access to Your NetSuite Account.

As part of a continuing commitment to provide the most secure application possible, since January 2015, we have been enforcing the prohibition against the use of iFrames on the following login pages:

- `/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 about defending against this vulnerability, visit the OWASP website to review the 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: Be aware of the following:

- 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).
- Oracle NetSuite does not support traffic that is routed through a split-tunnel Virtual Private Network (VPN) to control user access to NetSuite. For more information, see the help topic VPN Configuration for User Access to NetSuite.
- IP Address Rules are effective after successful login. The rules do not prevent a password reset or the login flow.

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 aware that you should use the two-factor authentication.

- 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.

SOAP web services, SAML Single Sign-on, and OpenID Connect 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 want to restrict access to your NetSuite account by employing IP address rules, see the following sections:

Note: The IP address rules can be also applied in sandbox account.

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).

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

To enable the IP address rules feature:

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**.

Important: 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.

To disable the IP address rules feature:

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

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 address 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:

   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, 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. 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 required search parameters in the available **User**, **Role**, and **IP Addresses Allowed to Login** fields.
   - For more information about 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 and OAuth 2.0 are the only mechanisms mature enough to use with web services and RESTlets.

Note: OAuth 2.0 cannot be used with SOAP web services. For more information, see OAuth 2.0.

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. The Three-Step TBA Authorization Flow should be used for all new integrations that are capable of opening a browser and handling a callback URL. Developers of existing integrations currently using the issuetoken endpoint should consider migrating the integration to the TBA authorization flow.

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, OpenID Connect (OIDC), and Two-Factor Authentication. You can use Two-Factor Authentication (2FA) roles and roles with SAML Single Sign-on permissions with TBA.

Note: Tokens created using the Token-based Authentication feature in your NetSuite production account are not copied to your Release Preview or to your sandbox accounts. To test this feature in Release Preview or in a sandbox, you must create new tokens in that account. Each time the sandbox is refreshed, you must create new tokens in the sandbox.

See the following topics for more information about TBA:

- Token-based Authentication (TBA) Tasks for Administrators
  - Getting Started with Token-based Authentication
  - Manage TBA Tokens in the NetSuite UI
  - Create Your Own Test Window for HMAC-SHA1 in TBA Integrations
- 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 (TBA) Tasks for Administrators

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

- Getting Started with Token-based Authentication
  - Enable the Token-based Authentication Feature

- Set Up Token-based Authentication Roles and Token-based Authentication (TBA) Permissions
- Assign Users to Token-based Authentication Roles
- Create Integration Records for Applications to Use TBA

- Manage TBA Tokens in the NetSuite UI
- Create Your Own Test Window for HMAC-SHA1 in TBA Integrations

Getting Started with Token-based Authentication

To set up token-based authentication (TBA) in your NetSuite account, you must complete the following tasks.

Click the links in the following steps for detailed instructions for each task.

To set up TBA in your NetSuite account:

1. Enable the Token-based Authentication Feature.
2. Set Up Token-based Authentication Roles.

   See also Token-based Authentication (TBA) Permissions.
3. Assign Users to Token-based Authentication Roles.
4. Create Integration Records for Applications to Use TBA.
5. Manage 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.

Enable 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 Manage Access Tokens 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 Create 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

Important: For enhanced security, two-factor authentication (2FA) is required 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 preferred, 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 cannot create access tokens for an administrator, and the administrator cannot create access tokens for another administrator.
  - **Cannot** create access tokens for their own use. **Exception:** administrators can create 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 **Manage Access Tokens** 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.

To add permissions to a role, go to Setup > Users/Roles > User Management > Manage Roles. Select a role to customize. On the Permission tab, Setup subtab, choose the permission from the list and click Add.

Note: A user assigned the User Access Tokens permission does not also need the Log in using Access Tokens permission.

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

Assign Users to Token-based Authentication Roles

After modifying roles with the appropriate token-based authentication permissions, an 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 about 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 Create Integration Records for Applications to Use TBA.

Create 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 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.

The following procedure briefly describes completing an integration record. You should create a separate integration record for each application.

To create an integration record for an application:

1. Go to Setup > Integration > Manage Integrations > New
2. Enter a **Name** for your application.
3. Enter a **Description**, if preferred.
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.
5. Enter a **Note**, if preferred. 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.

   In some cases, more than one method of authentication may be specified on an integration record.

> **Important:** You should transition from user credentials to another method of authentication. Specifying more than one method on a record can be useful when making the transition from user credentials to token-based authentication (TBA).

- For accessing SOAP web services, both the Token-based Authentication (TBA) and the User Credentials boxes can be checked.
- For accessing REST web services, both the Token-based Authentication (TBA) and the OAuth 2.0 boxes can be checked.
- For accessing RESTlets, the Token-based Authentication (TBA), the OAuth 2.0, and the User Credentials boxes can be checked.

Fields on the Authentication tab:	Effect when the box is checked:
Token-based Authentication (TBA)	■ This box must be checked to enable use of either the TBA authorization flow or the issuetoken endpoint. ■ When creating a new integration record, this box is checked by default. ■ Allows creation of tokens through the UI only. Use the tokens created to access RESTlets or SOAP and REST web services.
TBA: IssueToken Endpoint For more information, see The IssueToken Endpoint.	■ Allows programmatic creation of tokens using the issuetoken endpoint. ■ This box is checked for integration records that existed before your account was upgraded to 2019.2. **Important:** Check this box only if it is not possible to implement the TBA authorization flow in your integration.
TBA: Authorization Flow For more information, see The Three-Step TBA Authorization Flow.	■ When creating a new integration record, this box is checked by default. ■ Allows creation of tokens using the TBA authorization flow.
Callback URL	■ Enter the appropriate valid callback URL for your application. ■ The callback URL is validated when you save the integration record. **Note:** As of 2020.1, the callback URL supports multiple ports on a localhost (`http://localhost:*`). As of 2020.2, the callback URL supports using asterisk (*) as a part of a domain name. There are various ways to use the asterisk (*) in a domain name: ■ https://*.xyz.example.com/callback

Fields on the Authentication tab:	Effect when the box is checked:
	Following examples illustrate correct and incorrect callback URLs: ▫ **Correct:** https://myaccount.xyz.example.com/callback ▫ **Incorrect:** https://myaccount.prefix.xyz.example.com/callback ▫ **Incorrect:** https://myaccount.example.com/callback ■ https:// *.example.com/callback Following examples illustrate correct and incorrect callback URLs. ▫ **Correct:** https://myaccount.example.com/callback ▫ **Incorrect:** https://myaccount.prefix.example.com/callback ▫ **Incorrect:** https://example.com/callback You can use asterisk (*) as a first part of the domain name only.
User Credentials	**Important:** New integrations should use another method, such as TBA, rather than user credentials. ■ Clear this box to ensure this application will authenticate only using tokens and not with user credentials.

7. Click **Save**.

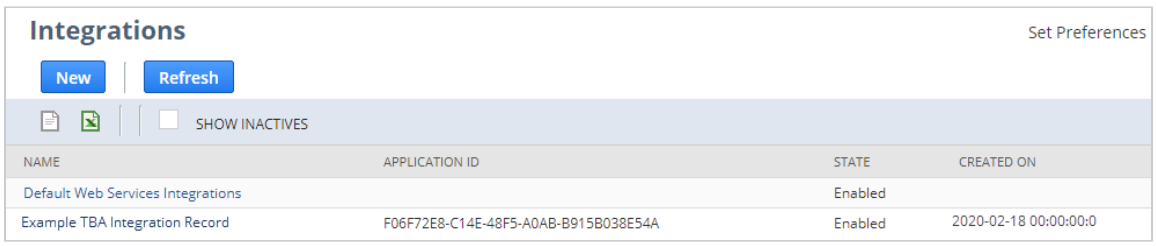
   The confirmation page displays the Client Credentials (Consumer Key and Consumer Secret) for this application. The application developer will need this information.

   Warning: The system displays the client ID and client secret only the first time you save the integration record. In cases where an application previously used user credentials as an authentication method, you must reset the consumer ID and consumer secret. Resetting the consumer ID and client secret invalidates the previous consumer ID and client secret.

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 Manage TBA Tokens in the NetSuite UI.

Important: Whether using The Three-Step TBA Authorization Flow, or 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 **Require Approval during Auto-Installation of Integration** box 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.

To view a list of integration records in this account, go to Setup > Integration > Integration Mangement > Manage Integrations.



Enabling an Existing 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 for SOAP web services, and that application might authenticate through user credentials. If appropriate, you can enable token-based authentication for that application.

To enable token-based authentication for an existing application:

1. Go 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 Client Credentials (Consumer Key and Consumer 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 client credentials (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.

Manage TBA Tokens in the NetSuite UI

Note: Tokens created using the Token-based Authentication feature in your NetSuite production account are not copied to your Release Preview or to your sandbox accounts. To test this feature in Release Preview or in a sandbox, you must create new tokens in that account. Each time the sandbox is refreshed, you must create new tokens in the sandbox.

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

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 can assign a token to those users who are assigned a role with only the **Log in using Access Tokens** permission. Administrators can create tokens for themselves, but not for other Administrators.

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.
3. On the Access Tokens page, click **New Access Token**.

   The Access token page appears.

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 preferred.
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.

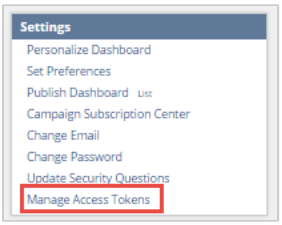
User Access Token – Create a TBA Token

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

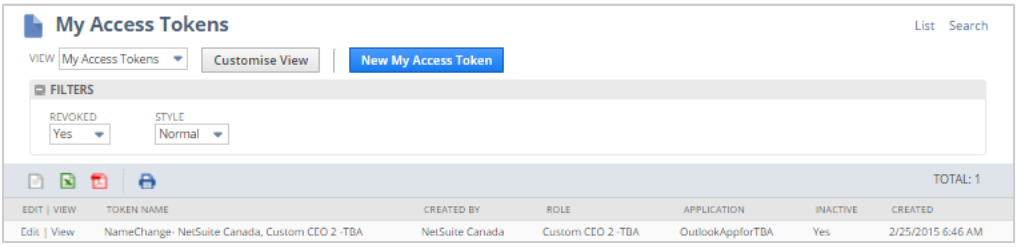
Note: Tokens created using the Token-based Authentication feature in your NetSuite production account are not copied to your Release Preview or to your sandbox accounts. To test this feature in Release Preview or in a sandbox, you must create new tokens in that account. Each time the sandbox is refreshed, you must 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 the **Manage Access Tokens** link.

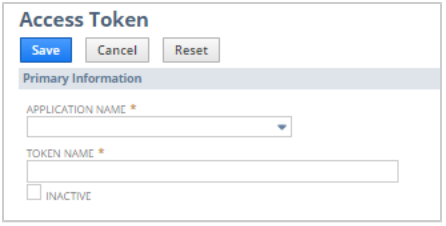


The My Access Tokens page appears, listing all the tokens for the current user in the current role.



3. Click **New My Access Token**.

   The Access Token page appears.



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 preferred.
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, theses 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.

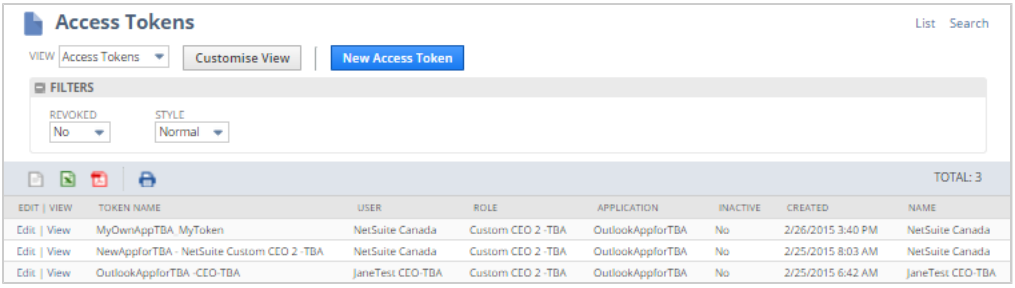
Viewing, Editing, Creating, and Revoking TBA Tokens

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

To view tokens:

1. Go to Setup > Users/Roles > User Management > Access Tokens

   The Access Token page appears.



2. 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

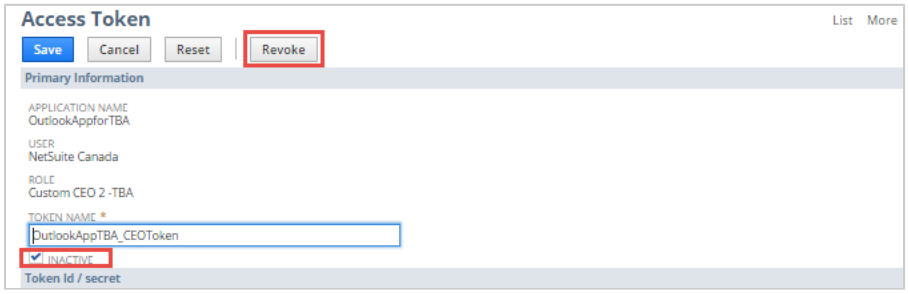
This section provides information about revoking a token in the NetSuite UI. For information about revoking a token programmatically, see Issue Token and Revoke Token REST Services for Token-based Authentication.

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.

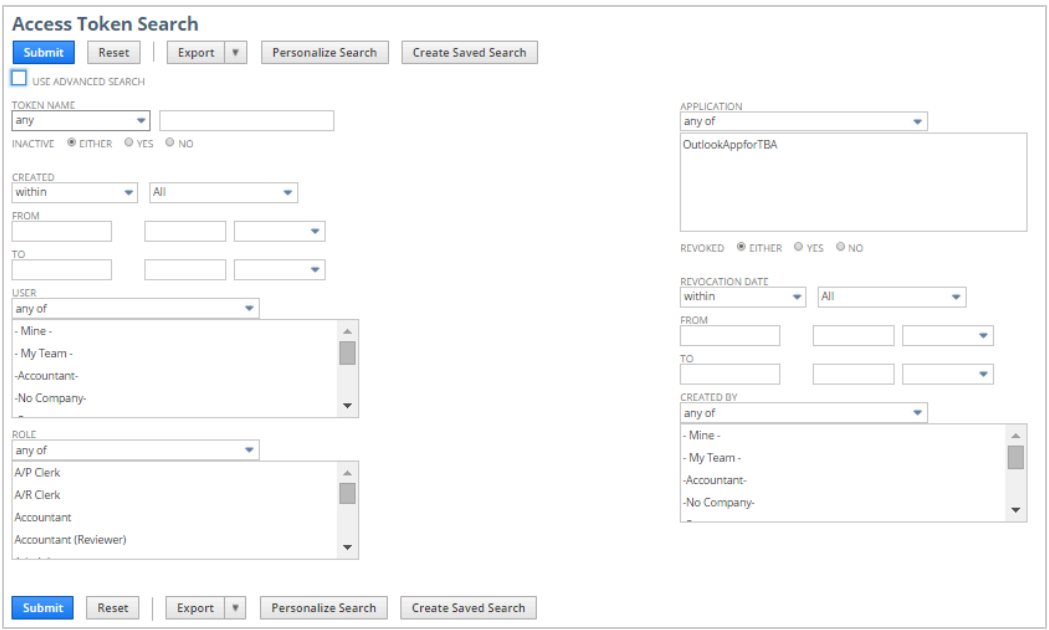
Using the TBA Access Token Search Page

There are two methods of opening the Access Token Search page. One method is to click **Search** 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:

1. Go to Setup > Users/Roles > Access Tokens > Search.

   The Access Token Search page appears.



2. Enter or select from the available criteria, as appropriate.
3. click **Submit**.

For information about NetSuite's search capabilities, see:

- Running Searches
- Saved Searches

Create Your Own Test Window for HMAC-SHA1 in TBA Integrations

Using the HMAC-SHA1 signature method for Token-based Authentication (TBA) integrations is no longer considered best practice. To help you prepare for the end of support in the future, you can temporarily disable the HMAC-SHA1 signature method for TBA in your production account. Disabling HMAC-SHA1 temporarily lets you test your TBA integrations to verify that they are not using the HMAC SHA1 signature method.

Note: Support ended on July 30, 2021 for the HMAC-SHA1 signature method for TBA integrations in all non-production accounts, such as sandbox, Test Drive, development accounts, and Release Preview accounts.

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 temporarily disable HMAC-SHA1 for TBA in your NetSuite production account:

1. Go to Setup > Company > Enable Features.
2. Click the **SuiteCloud** tab, and scroll down to the Manage Authentication section.
3. Check the **Disable HMAC-SHA1 for Token-based Authentication** box.
4. Click **Save**.

To re-enable HMAC-SHA1 for TBA, clear the **Disable HMAC-SHA1 for Token-based Authentication** box, and click **Save**.

Warning: The end of support for the HMAC-SHA1 signature method is targeted for 2023.1. You should update your integrations to use the HMAC-SHA256 signature method as soon as possible.

For more information, see The Signature for Web Services and RESTlets.

Token-based Authentication (TBA) for Integration Application Developers

Developers now have options for granting tokens for applications. If you decide to use TBA for new integrations, you should use the TBA authorization flow. Developers of existing integrations currently using the issuetoken endpoint should consider migrating the integration to the TBA authorization flow.

See the following for more information about these options:

The Three-Step TBA Authorization Flow

As of 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. Users enter user credentials into one of the following login forms as a part of the flow:

- A trusted NetSuite login form.
- SAML SSO identity provider's login form
- OIDC OP provider's login form.

The redirection-based authorization flow consists of three steps. Click the following links for more detailed information about each step.

- Step One Obtain An Unauthorized Request Token on the request token URL.
- Step Two 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 Three Exchange the Request Token for an Access Token on the access token URL.

With the TBA authorization flow, 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 can access NetSuite data. If this step succeeds, the application exchanges the request token for an access token to be used when calling a RESTlet or a web service.

If you decide to use TBA for new integrations, you should use the TBA authorization flow. Developers of existing integrations currently using the issuetoken endpoint should consider migrating the integration to the TBA authorization flow.

The administrator must create integration records for each application. See Create 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 and Issue Token and Revoke Token REST Services for Token-based Authentication. A tokeninfo endpoint is also available to provide information about a user based on the access token. See Calling a token endpoint to obtain user information based on a token.

Step One Obtain An Unauthorized Request Token

The application sends a POST request to the request token endpoint. Include the necessary parameters in the authorization header.

The format of the URL is:

```
https://<accountID>.restlets.api.netsuite.com/rest/requesttoken
```

where <accountID> is a variable for your NetSuite account ID.

Note: You should use the account-specific domain URL as shown. However, as of 2020.1, if you do not know the account ID, requests can be sent to the `system.netsuite.com` domain.

See the following header for details.

Request Header Parameters in the Authorization Header for Step One

OAuth Authorization Header Parameter	Description
oauth_consumer_key	■ Identifies the client. (The service attempting to access the resource.) ■ The value of the consumer key is provided when the integration record is created.
oauth_signature_method	Only HMAC-SHA256 is supported.
oauth_signature	■ Constructed signature (consumer secret to be used during signing) For more information about constructing a signature, see Constructing the Signature for Step One of the TBA Authorization Flow. See also Specifications for Signature Construction for the TBA Authorization Flow.
oauth_timestamp	■ Number of seconds passed since 1st January 1970 00:00:00 GMT ■ Must be a positive integer ■ Should be equal to or greater than any timestamp passed in previous requests
oauth_nonce	■ Generated random string. Nonce must be at least six characters long. An ideal nonce length is 20 characters. ■ Must be unique for all requests with the same timestamp.
oauth_version	■ Optional. ■ If present, value must be `1.0`.
oauth_callback	■ An absolute URL, to which a redirect with a verification code will be performed. ■ The callback URL should match the callback URL in the corresponding integration record. ■ As of 2020.1, the callback URL supports multiple ports on a localhost (`http://localhost:*`). This is the only case where use of the asterisk (*) character is permitted.
realm	■ NetSuite account ID (company identifier). **Note:** As of 2020.1, the realm parameter is no longer required for this step.
role	■ Optional. ■ Indicates the role for which to grant the access token.
state	■ Optional. ■ Maximum length is 512 characters. Valid alpha-numeric characters are upper- and lowercase letters (a-z, A-Z), and numbers 0–9.

OAuth Authorization Header Parameter	Description
	Refer to RFC 6749, Section 4.1.1 for more information about the state parameter.

Note: Refer to RFC 5849 if you need more information about the parameters oauth_timestamp, oauth_nonce, and oauth_version.

The HTTP Response Parameters for Step One

When an authorization request is successfully verified, the following HTTP response is returned:

Response Parameter	Description
oauth_token	An unauthorized request token, which should be authorized by the application in Step Two of the flow.
oauth_token_secret	The corresponding token secret, to be used for signature creation in Step Three of the flow.
oauth_callback_confirmed	Response must be true, if the request verification was successful.
role	The role parameter is present in the response only if configured in the request.
state	The state parameter is present in the response only if configured in the request. The value of the parameter must match the value in the request.

When you have the HTTP response, proceed to Step Two Authorize the Request Token.

Step Two Authorize the Request Token

The application sends a GET request to the user authorization endpoint. Include the oauth_token parameter obtained in the response in Step One.

The format of the URL is:

```
https://<accountID>.app.netsuite.com/app/login/secure/authorizetoken.nl?
oauth_token=da9eba68ac7c1995bcdcb5f035f5b64df79dbc6e4db305064aa63eaa7bf35111
```

where <accountID> is a variable for your NetSuite account ID.

Note: You should use the account-specific domain URL as shown. However, as of 2020.1, if you do not know the account ID, requests can be sent to the `system.netsuite.com` domain.

- The user is authenticated. If there is no active NetSuite session, the user is first redirected to the NetSuite login form. If the GET request points to an account-specific domain, for an account with SAML SSO or OIDC enabled, the user can be redirected to a third party application.
- After successful authentication, a consent page appears. The user can click **Allow** to give permission for the generation of the access token, which occurs in Step Three.

  Note: If the user clicks **Deny**, the authorization flow ends. The application should display an error message to the user. Clicking **Deny** is one reason for an empty `oauth_verifier` parameter in the response to Step Two.

- If the authenticated user is logged in to an inappropriate role, the user can choose the appropriate role by selecting **Change Role** on the Consent page.

Redirect Parameters for Step Two

The user is redirected to the `oauth_callback` URL (from Step One), with the `oauth_token` and the `oauth_verifier` parameters.

The following is an example of a redirect:

```
https://my.example.com/TBA/?callbackRequest&oauth_token=da9eba68ac7c1995bcdcb5f035f5b64df79dbc6e4db305064aa63eaa7bf35111&oauth_verifier=111e630079c0222cf59cf18410e9939c848507457d7010003db01e63fa42abcd&company=1234567&role=3&entity=38
```

Parameter	Description
oauth_token	An authorized request token to be used in Step Three.
oauth_verifier	An attribute to be used in Step Three.
company	NetSuite account ID (company identifier).
role	Indicates the role for which to grant the access token.
entity	The entity ID of a successfully authenticated system user.
state	If the optional state parameter value does not match the value originally passed to NetSuite, the client should not trust the request or redirect.

When the application has handled the callback URL, proceed to Step Three: Step Three Exchange the Request Token for an Access Token.

Step Three Exchange the Request Token for an Access Token

The application should send a POST request to the access token endpoint. Include the necessary parameters in the authorization header.

The format of the URL is:

`https://<accountID>.restlets.api.netsuite.com/rest/accesstoken`

where <accountID> is a variable for your NetSuite account ID.

Request Header Parameters in the Authorization Header for Step Three

OAuth Authorization Header Parameter	Description
oauth_consumer_key	The same verified oauth_consumer_key value that was used in Step One, from the Integration record.
oauth_token	The authorized request token from the response in Step Two.
▪ oauth_signature_method ▪ oauth_timestamp ▪ oauth_nonce ▪ oauth_version	▪ Only HMAC-SHA256 is supported for the signature method. ▪ Should be equal to or greater than any timestamp passed in previous requests. ▪ Nonce must be unique for all requests with the same timestamp. Length should be 20 characters. ▪ oauth_version is optional, but if present, must be `1.0`.

OAuth Authorization Header Parameter	Description
oauth_verifier	The attribute from Step Two.
oauth_signature	Similar to the procedure in Step One, but also including the token secret which was returned in Step One. For more information about constructing a signature, see Constructing the Signature for Step Three of the TBA Authorization Flow. See also Specifications for Signature Construction for the TBA Authorization Flow.
realm	NetSuite account ID (company identifier). **Note:** As of 2020.1, the realm parameter is no longer required for this step.

Important: Whether using The Three-Step TBA Authorization Flow, or 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 **Require Approval during Auto-Installation of Integration** box 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.

Response Parameters for Step Three

Response Parameter	Description
▪ oauth_token ▪ oauth_token_secret	A granted access token and token secret to be used for proper authorization header compilation to call a RESTlet or a web service. For more information, see The Authorization Headers.

If the access token is generated successfully, the integration record is automatically installed for the requested account. For more information, see the help topic Auto-Installation of Integration Records.

Specifications for Signature Construction for the TBA Authorization Flow

This section contains details about the specifications for creating signatures required for both Step One and Step Three of the TBA authorization flow. For more information about signatures, refer to Section 3.4 of RFC 5849.

Warning: The end of support for the HMAC-SHA1 signature method is targeted for 2023.1. You should update your integrations to use the HMAC-SHA256 signature method as soon as possible.

Encoding

For more information about encoding, refer to Section 3.6 of RFC 5849:

- For **Text values**, refer to RFC 3629. Text values must be encoded as UTF-8 octets if they are not already encoded.
- Values are escaped using the **Percent-Encoding** (%XX) mechanism:
  - Do not encode characters from the unreserved character set. Refer to Section 2.3 of RFC 3986 for documentation of the unreserved character set.
  - All other characters must be encoded.
  - Two hexadecimal characters used to represent encoded characters must be uppercase.

> **Important:** A blank symbol is encoded as %20 and not as the plus (+) symbol. Be aware that some framework functions may return unwanted results.

Request Parameters Normalization

For more information, refer to Section 3.4.1.3.2 of RFC 5849.

- The parameters that are used include: (refer to Request Parameters, Section 3.4.1.3 of RFC 5849):
  - parameters from the Authorization header (excluding “realm” and “oauth_signature”)
  - parameters from the HTTP request entity body
  - parameters from the query part of the request URL
- Encoding of parameter names and values occurs using the algorithm described in Encoding.
- Sorting by name is performed using ascending byte value ordering. If names are identical, sorting is done by values.
- Names and values form pairs separated by the equal (=) symbol, even when there is no value.
- Pairs are concatenated in the defined order by the ampersand (&) symbol.

Generating the Signature for the TBA Authorization Flow

This section contains details about generating the signature required for both Step One and Step Three of the TBA authorization flow.

The following example is showing the way the signature should be constructed. The final result depends on the language you use for generating the signature.

```
signature = HMAC-SHA256(key, text)
```

Where:

- The value of the text parameter is the base string from the appropriate section:
  - Signature Base String Construction for Step One
  - Generating the Signature for Step Three
- The value of the key parameter is the concatenation—using the ampersand (&) character—of the consumer secret and the token secret with both values encoded by the algorithm described in Encoding.

> **Important:** The token secret value is only used in Step Three. The token secret value is empty in Step One.

The result digest octet string is used as the resulting oauth_signature parameter after:

- being Base64-encoded. (For more information about Base64 Content-Transfer-Encoding, see Section 6.8 of RFC 2045.
- being encoded using the algorithm described in Encoding.

Warning: The end of support for the HMAC-SHA1 signature method is targeted for 2023.1. You should update your integrations to use the HMAC-SHA256 signature method as soon as possible.

For more information, see the following topics

- Constructing the Signature for Step One of the TBA Authorization Flow
- Constructing the Signature for Step Three of the TBA Authorization Flow

Constructing the Signature for Step One of the TBA Authorization Flow

This section contains information and examples for how to construct the signature used in Step One of the TBA authorization flow.

Warning: The end of support for the HMAC-SHA1 signature method is targeted for 2023.1. You should update your integrations to use the HMAC-SHA256 signature method as soon as possible.

The following values are used for the examples in this section:

Parameter	Value
Company ID	1234567
Role ID	45678
Consumer Key	60712990bc09623786e7047c226bcb3f86d49dca0b04efc21001dc76d97a81f5
Consumer Secret	60712990bc09623786e7047c226bcb3f86d49dca0b04efc21001dc76d97a81f5 **Note:** For purposes of this example, the values of Consumer Key and Consumer Secret are identical.
Nonce	bUvpxBX93OWo0FLswq5M
Timestamp	1575998103
Callback URL	https://my.example.com/TBA/?callbackRequest

Signature Base String Construction for Step One

The formation for the construction of the base string is as follows:

```
<base-string> = <http-request-method>&<base-string-uri>&<normalized-request-parameters>
```

Where:

Component	Description
http-request-method	POST
base-string-uri	https://1234567.restlets.api.netsuite.com/rest/requesttoken

Component	Description
	Note: The URI is to be encoded using the algorithm described in Encoding.
normalized-request-parameters	The following parameters to be normalized into a single string are: ■ oauth_callback ■ oauth_consumer_key ■ oauth_nonce ■ oauth_signature_method ■ oauth_timestamp ■ oauth_version ■ role **Note:** The single string of normalized parameters is to be encoded using the algorithm described in Request Parameters Normalization.

Signature Base String Example for Step One

```
POST&https%3A%2F%2F1234567.restlets.api.netsuite.com%2Frest%2Frequesttoken&oauth_callback%3Dhttps%253A%252F%252Fmy.example.com%252FTBA%252F%253FcallbackRequest%26oauth_consumer_key%3D60712990bc09623786e7047c226bcb3f86d49dca0b04efc21001dc76d97a81f5%26oauth_nonce%3DbUvpxBX93OWo0FLswq5M%26oauth_signature_method%3DHMAC-SHA256%26oauth_timestamp%3D1575998103%26oauth_version%3D1.0%26role%3D45678
```

Generating the Signature for Step One

The key for generating the signature consists of the consumer secret.

Important: Be aware that the token secret is omitted in Step One.

```
60712990bc09623786e7047c226bcb3f86d49dca0b04efc21001dc76d97a81f5&
```

After using the algorithm described in Generating the Signature for the TBA Authorization Flow you get the following result:

```
7kgwwmiAylqeMdHjCBnIUUW%2BdrDrGCbZGBkuCt39J90%3D
```

Final Authorization Header Example for Step One

```
Authorization: OAuth realm="1234567", role="45678", oauth_consumer_key="60712990bc09623786e7047c226bcb3f86d49dca0b04efc21001dc76d97a81f5", oauth_nonce="bUvpxBX93OWo0FLswq5M", oauth_timestamp="1575998103", oauth_signature_method="HMAC-SHA256", oauth_version="1.0", oauth_callback="https%3A%2F%2Fmy.example.com%2FTBA%2F%3FcallbackRequest", oauth_signature="7kgwwmiAylqeMdHjCBnIUUW%2BdrDrGCbZGBkuCt39J90%3D"
```

Constructing the Signature for Step Three of the TBA Authorization Flow

This section contains information and examples for how to construct the signature used in Step Three of the TBA authorization flow.

Warning: The end of support for the HMAC-SHA1 signature method is targeted for 2023.1. You should update your integrations to use the HMAC-SHA256 signature method as soon as possible.

The following values are used for the examples in this section:

Parameter	Value
Company ID	1234567
Consumer Key	60712990bc09623786e7047c226bcb3f86d49dca0b04efc21001dc76d97a81f5
Consumer Secret	60712990bc09623786e7047c226bcb3f86d49dca0b04efc21001dc76d97a81f5 **Note:** For purposes of this example, the values of Consumer Key and Consumer Secret are identical.
Token Key	447d0cba5569a2d616e32a537110bc8c10ebcf42cc1fa34d6f76d08531abc179
Token Secret	447d0cba5569a2d616e32a537110bc8c10ebcf42cc1fa34d6f76d08531abc179 **Note:** For purposes of this example, the values of Token Key and Token Secret are identical.
Verifier	3eff1ae4de6f924014b88e489a41e88da8ed1ba8bd5ad7684a71579d7e97f4ee
Nonce	wjRgXQPWhYtKl0A7bO8Z
Timestamp	1576079512

Important: The realm parameter is not a part of the single string. For more information, see RFC5849 section 3.4.1.3.1.

Signature Base String Construction for Step Three

The formation for the construction of the base string is as follows:

```
<base-string> = <http-request-method>&<base-string-uri>&<normalized-request-parameters>
```

Where:

Component	Description
http-request-method	POST
base-string-uri	https://1234567.restlets.api.netsuite.com/rest/accesstoken **Note:** The URI is to be encoded using the algorithm described in Encoding.
normalized-request-parameters	The following parameters to be normalized into a single string are: - oauth_consumer_key - oauth_token - oauth_signature_method - oauth_timestamp - oauth_nonce - oauth_version - oauth_verifier

Component	Description
	Note: The single string of normalized parameters is to be encoded using the algorithm described in Request Parameters Normalization.

Signature Base String Example for Step Three

```
POST&https%3A%2F%2F1234567.restlets.api.netsuite.com%2Frest%2Faccesstoken&oauth_con
sumer_key%3D60712990bc09623786e7047c226bcb3f86d49dca0b04efc21001dc76d97a81f5%26oauth_nonce%3DwjRgXQPWhYtKl0A7bO8Z%26oauth_sig
nature_method%3DHMAC-SHA256%26oauth_timestamp%3D1576079512%26oauth_token%3D447d0cba5569a2d616e32a537110bc8c10ebcf42c
c1fa34d6f76d08531abc179%26oauth_verifier%3D3eff1ae4de6f924014b88e489a41e88da8ed1ba8bd5ad7684a71579d7e97f4ee%26oauth_version%3D1.0
```

Generating the Signature for Step Three

The key for generating the signature consists of the consumer secret and the token secret.

Important: Be aware that the token secret is present in Step Three, whereas it was omitted in Step One.

```
60712990bc09623786e7047c226bcb3f86d49dca0b04efc21001dc76d97a81f5&447d0cba5569a2d616e32a537110bc8c10ebcf42cc1fa34d6f76d08531abc179
```

After using the algorithm described in Generating the Signature for the TBA Authorization Flow you get the following result:

```
BBzawyjesZyFrwBjUAJfBsPDDGUY2FRdp3k4NwGDAO0%3D
```

Final Authorization Header Example for Step Three

```
Authorization: OAuth realm="1234567", oauth_token="447d0cba5569a2d616e32a537110bc8c10ebcf42cc1fa34d6f76d08531abc179",
 oauth_consumer_key="60712990bc09623786e7047c226bcb3f86d49dca0b04efc21001dc76d97a81f5", oauth_nonce="wjRgXQPWhYtK
l0A7bO8Z", oauth_timestamp="1576079512", oauth_signature_method="HMAC-SHA256", oauth_version="1.0", oauth_verifier="3ef
f1ae4de6f924014b88e489a41e88da8ed1ba8bd5ad7684a71579d7e97f4ee", oauth_signature="BBzawyjesZyFrwBjUAJfBsPDDGUY2FRdp3k4NwGDAO0%3D"
```

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.

If you decide to use TBA for new integrations, you should use the TBA authorization flow. Developers of existing integrations currently using the issuetoken endpoint should consider migrating the integration to the authorization flow. See The Three-Step TBA Authorization Flow for more information.

Important: Whether using The Three-Step TBA Authorization Flow, or 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 **Require Approval during Auto-Installation of Integration** box 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.

See the following sections for more information:

Required 2FA, the IssueToken Endpoint, and nlauth_otp

Important: See The Three-Step TBA Authorization Flow, which should be used for all new integrations that use TBA. Developers of existing integrations currently using the issuetoken endpoint should consider migrating the integration to the TBA authorization flow.

To accommodate the requirement for required 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 to review the standard.

Warning: If the authenticator app is on the same device you use to access the integrations, it is not considered secure. Similarity to the 2FA verification code can only be maintained if the authenticator and the integration are not on the same device.

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 one–time password (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

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 use various 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 about 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

See The Three-Step TBA Authorization Flow, which should be used for all new integrations that use TBA. Developers of existing integrations currently using the NLAuth authorization header should consider migrating the integration to the TBA authorization flow.

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.

Field	Description
nlauth_account	The account ID of the NetSuite account.
nlauth_email	The email address with which the user logs in to NetSuite.
nlauth_signature	The user's password.
nlauth_role	The internal ID of a role with which the user is associated.
nlauth_application_id	The application ID of the integration associated with the RESTlet.
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.

Field	Description
Important: This parameter can only be used with the issuetoken endpoint.	

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

Token-based Authentication (TBA) for Users

For users without a role that has the User Access Token permission, an administrator can create, assign, and manage access tokens.

Users also have the following options to obtain their own access tokens:

- Users assigned a role that has the User Access Token permission can create, assign, and manage tokens for the current user and current role. For more information, see User Access Token – Create a TBA Token
- An integration developer creates an application that requests user credentials and gives the user an access token. For more information, see The IssueToken Endpoint.
- An integration developer creates an application that uses the TBA authorization flow. At the end of the flow, user gets an access token. For more information, see The Three-Step TBA Authorization Flow.

Note: Tokens created using the Token-based Authentication feature in your NetSuite production account are not copied to your Release Preview or to your sandbox accounts. To test this feature in Release Preview or in a sandbox, you must create new tokens in that account. Each time the sandbox is refreshed, you must create new tokens in the sandbox.

Troubleshoot Token-based Authentication (TBA)

Note: Tokens created using the Token-based Authentication feature in your NetSuite production account are not copied to your Release Preview or to your sandbox accounts. To test this feature in Release Preview or in a sandbox, you must create new tokens in that account. Each time the sandbox is refreshed, you must create new tokens in the sandbox.

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
    - Error Messages for RESTlets, SOAP Web Services, and REST Web Services
    - Error Messages for the TBA Authorization Flow
- The Signature for Web Services and RESTlets
  - Generate a Signature
    - Input Parameters for the Example
    - Step One Construct a Base String for the Signature

Note: Encoding used in TBA is percent encoding. All code examples in the listed sections use rawurlencoding, the PHP implementation of percent encoding. For more information, see the specification https://tools.ietf.org/html/rfc5849#section-3.6.

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 and REST web services have slightly different error messages, but the meaning is similar.

For more information about error messages, see the following sections:

Note: Tokens created using the Token-based Authentication feature in your NetSuite production account are not copied to your Release Preview or to your sandbox accounts. To test this feature in Release Preview or in a sandbox, you must create new tokens in that account. Each time the sandbox is refreshed, you must create new tokens in the sandbox.

Error Messages for RESTlets, SOAP Web Services, and REST Web Services

See the following table for information about resolving error messages for RESTlets, SOAP web services, and REST web services.

RESTlets	SOAP and REST Web Services	Problem	Resolution
consumer_key_ refused	consumer_key_ refused	The application is in **Blocked** state on the integration record.	Enable the application on the integration record. **To enable the app:** 1. Go to Setup > Integration > Manage Integrations. 2. Select the appropriate integration record, and click **Edit**. 3. In the **State** field, change **Blocked** to **Enabled**. 4. Save the record.
consumer_key_ unknown	consumer_key_ unknown	The appropriate integration record could not be found.	▪ Ensure the consumer key is correct. ▪ If this error occurs with a currently enabled application, you can attempt resetting the credentials (obtain new credentials). **Important:** 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 Create Integration Records for Applications to Use TBA.
FeatureDisabled	FeatureDisabled	The Token-based Authentication feature in NetSuite is not enabled.	Enable the feature. See Enable the Token-based Authentication Feature.
MissingToken PassportRequired Fields	MissingToken PassportRequired Fields	The request is missing a required parameter.	Verify that all required parameters are included in the request.
nonce_rejected	—	The nonce was not long enough.	Nonce must be at least six characters long. An ideal nonce length is 20 characters.
nonce_used	NonceUsed	The combination of nonce and timestamp has already been used by this user.	▪ 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.
parameter_rejected	—	The parameter was either: ▪ sent twice. ▪ sent with a malformed value. ▪ sent with an empty value.	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.

RESTlets	SOAP and REST Web Services	Problem	Resolution
permission_denied	permission_denied	The entity or role is not usable.	This error may have many causes. ■ Verify that the entity and role are both active in NetSuite. ■ Verify the entity has access. ■ Verify that the role has TBA permissions. ■ Verify that the user has not made the role inactive on the user's View My Roles page.
InvalidSignature	InvalidSignature	The request was not signed correctly.	See Generate a Signature for the correct method of signing a request.
UnknownAlgorithm	UnknownAlgorithm	The algorithm used to create signature is not supported.	The most secure supported algorithm is HMAC-SHA256. You should use the HMAC-SHA256 algorithm. ■ Currently, the HMAC-SHA1 algorithm is supported for the issuetoken endpoint, but HMAC-SHA256 is preferred. **Warning:** The end of support for the HMAC-SHA1 signature method is targeted for 2023.1. You should update your integrations to use the HMAC-SHA256 signature method as soon as possible. **Note:** Only the HMAC-SHA256 algorithm is supported for the TBA authorization flow. See Error Messages for the TBA Authorization Flow.
temporary_locked	temporary_locked	The user is locked out of NetSuite.	The user was locked out of NetSuite after five failed login attempts. The user must wait 30 minutes to unlock access to your account. Or, the user can ask their administrator or system administrator for a password reset. See User Access Reset Tool.
InvalidTimestamp	InvalidTimestamp	The timestamp of the request must be within plus or minus five (+ or – 5) minutes of the server time.	Ensure that: ■ Your computer clocks are synchronized using the NTP protocol. ■ Requests are sent soon after generating the TBA header. ■ Requests are not being queued before being sent to NetSuite.
token_rejected	token_rejected	The token could not be found.	Ensure that the token: ■ Is correct. ■ Is active. ■ Is a token for the correct integration application. If a token does not exist, create one. See Manage TBA Tokens in the NetSuite UI.
VersionRejected	—	The request uses an invalid value for OAuth version parameter.	The value for the OAuth version parameter must be 1.0.

Error Messages for the TBA Authorization Flow

See the following table for information about resolving error messages for the TBA authorization flow.

Request Token Errors	Access Token Errors	Problem	Resolution
UnknownIntegration	UnknownIntegration	The appropriate integration record could not be found.	■ If the integration record exists, verify the following:

Request Token Errors	Access Token Errors	Problem	Resolution
			▫ Ensure the consumer key is correct. ▫ If this error occurs with a currently enabled application, you can attempt resetting the credentials (obtain new credentials). **Important:** 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 Create Integration Records for Applications to Use TBA.
— **Note:** If the company ID parameter is specified in Step 1 of the TBA authorization flow, the value of the error is IntegrationBlocked.	IntegrationBlocked	The state of the integration record is **Blocked.**	Enable the application on the integration record. **To enable the app:** 1. Go to Setup > Integration > Manage Integrations. 2. Select the appropriate integration record, and click **Edit**. 3. In the **State** field, change **Blocked** to **Enabled**. 4. Save the record.
AuthorizationFlowRequired	AuthorizationFlow Required	The integration application does not use the TBA authorization flow.	Ensure that the **TBA: Authorization Flow** box is checked in the corresponding integration record. For more information, see Create Integration Records for Applications to Use TBA.
InvalidTimestamp	InvalidTimestamp	The timestamp is not in the allowed range. The timestamp of the request must be within plus or minus five (+ or – 5) minutes of the server time.	Ensure that: ▪ Your computer clocks are synchronized using the NTP protocol. ▪ Requests are sent soon after generating the TBA header. ▪ Requests are not being queued before being sent to NetSuite.
InvalidSignature	InvalidSignature	The signature is incorrect.	See Generating the Signature for the TBA Authorization Flow for the correct method of signing a request. See also The Signature for Web Services and RESTlets.
InvalidCallback <<--callback-URL-specified-->>	—	The callback URL is not valid.	On the integration record, verify the callback URL and correct the request as needed.
MissingRequiredParameter	MissingRequired Parameter	The request is missing a required parameter.	Verify that all required parameters are included in the request. For more information, see the following topics: ▪ Step One Obtain An Unauthorized Request Token for Step One of the TBA flow. ▪ Step Three Exchange the Request Token for an Access Token for Step Three of the TBA flow.

Request Token Errors	Access Token Errors	Problem	Resolution
NonceRejected	NonceRejected	The nonce was not long enough.	Nonce must be at least six characters long. An ideal nonce length is 20 characters.
NonceUsed	NonceUsed	The combination of nonce and timestamp has already been used by this user.	■ 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.
—	TokenRejected	The token could not be found.	Ensure that the token: ■ Is correct. ■ Is active. ■ Is a token for the correct integration application.
—	EntityOrRoleDisabled	The entity or role is not usable.	This error may have many causes. ■ Verify that the entity and role are both active in NetSuite. ■ Verify the entity has access. ■ Verify that the role has TBA permissions. ■ Verify that the user has not made the role inactive on the user's View My Roles page.
FeatureDisabled	FeatureDisabled	The Token-based Authentication feature in NetSuite is not enabled.	Enable the feature. See Enable the Token-based Authentication Feature.
UnknownAlgorithm	UnknownAlgorithm	Potential reasons for this error message include: ■ The algorithm used (such as HMAC-SHA1) is not supported for the TBA authorization flow. ■ The signature method is not supported.	Potential resolutions include: ■ Only the HMAC-SHA256 algorithm is supported for the TBA authorization flow ■ See Generating the Signature for the TBA Authorization Flow for the correct method of signing a request.
VersionRejected	VersionRejected	The request uses an invalid value for OAuth version parameter.	The value for the OAuth version parameter must be 1.0.
—	InvalidVerifier	The value of the oauth_verifier parameter does not match the value provided in Step Two of the TBA flow.	Ensure that the value of the `oauth_verifier` parameter provided to the Callback URL in the application is used during Step Three.

The Signature for Web Services and RESTlets

This section covers generating a valid signature. The examples shown are for SOAP web services, REST web services, and for RESTlets. The principle for constructing a signature is similar for the TBA authorization flow. The TBA authorization flow requires additional parameters that are not shown in the following examples. For more information about the required parameters, see The Three-Step TBA Authorization Flow.

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

Generate a Signature

Warning: The end of support for the HMAC-SHA1 signature method is targeted for 2023.1. You should update your integrations to use the HMAC-SHA256 signature method as soon as possible.

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 One Construct a Base String for the Signature
- Step Two Signature Key
- Step Three 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.

```
$url = 'https://123456.restlets.api.netsuite.com/app/site/hosting/restlet.nl?script=6&deploy=1&customParam=someValue&test
Param=someOtherValue';
//or https://123456.suitetalk.api.netsuite.com/services/NetSuitePort_2015_2 for webservices
//or https://123456.suitetalk.api.netsuite.com/services/rest/record/v1/employee/40 for REST web services
$httpMethod = 'POST'; //or $httpMethod = 'GET'; for REST Web Services
$tokenKey = '2b0ce516420110bcbd36b69e99196d1b7f6de3c6234c5afb799b73d87569f5cc';
$tokenSecret = 'c29a677df7d5439a458c063654187e3d678d73aca8e3c9d8bea1478a3eb0d295';
$consumerKey = 'ef40afdd8abaac111b13825dd5e5e2ddddb44f86d5a0dd6dcf38c20aae6b67e4';
$consumerSecret = 'd26ad321a4b2f23b0741c8d38392ce01c3e23e109df6c96eac6d099e9ab9e8b5';
$signatureMethod = 'HMAC-SHA256';
$nonce = 'fjaLirsIcCGVZWzBX0pg';      //substr(str_shuffle("0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"), 0,
 20);
$timestamp = '1508242306';            //time();
$version = '1.0';
$realm = '123456';                   //scompid
```

Step One 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.

Note: If you are constructing a signature for the TBA authorization flow, be aware of the following:

- The token and oauth_verifier parameters required for the base string are not shown in the following examples. See The Three-Step TBA Authorization Flow for information about these parameters.
- Except for the realm parameter, all parameters shown in the table in Request Header Parameters in the Authorization Header for Step One must be part of base string.
- You can follow the RESTlets format as a guideline for constructing the base string, as RESTlets also follows the OAuth 1.0 specification.

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.

```
123456&ef40afdd8abaac111b13825dd5e5e2ddddb44f86d5a0dd6dcf38c20aae6b67e4&2b0ce516420110bcbd36b69e99196d1b7f6de3c6234c5af
b799b73d87569f5cc&fjaLirsIcCGVZWzBX0pg&1508242306
```

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%3A%2F%2F123456.restlets.api.netsuite.com%2Fapp%2Fsite%2Fhosting%2Frestlet.nl%customParam%3DsomeValue%26deploy
%3D1%26oauth_consumer_key%3Def40afdd8abaac111b13825dd5e5e2ddddb44f86d5a0dd6dcf38c20aae6b67e4%26oauth_nonce%3DfjaLirsIc
CGVZWzBX0pg%26oauth_signature_method%3DHMAC-SHA256%26oauth_timestamp%3D1508242306%26oauth_token%3D2b0ce516420110bcb
d36b69e99196d1b7f6de3c6234c5afb799b73d87569f5cc%26oauth_version%3D1.0%26script%3D6%26testParam%3DsomeOtherValue
```

REST Web Services

```
$baseString = oauth_get_sbs($httpMethod, $url, array('oauth_consumer_key' => $consumerKey,
                                                     'oauth_nonce' => $nonce,
                                                     'oauth_signature_method' => $signatureMethod,
```

```
                                                'oauth_timestamp' => $timestamp,
                                                'oauth_token' => $tokenKey,
                                                'oauth_version' => $version));
```

Example 3. REST Web Services Base String Example

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

```
GET&https%3A%2F%2F123456.suitetalk.api.netsuite.com%2Fservices%2Frest%2Frecord%2Fv1%2Femployee%2F40&oauth_consumer_key%3Def40afdd8abaac111b13825dd5e5e2ddddb44f86d5a0dd6dcf38c20aae6b67e4%26oauth_nonce%3DfjaLirsIcCGVZWzBX0pg%26oauth_signature_method%3DHMAC-SHA256%26oauth_timestamp%3D1508242306%26oauth_token%3D2b0ce516420110bcbd36b69e99196d1b7f6de3c6234c5afb799b73d87569f5cc%26oauth_version%3D1.0
```

Step Two 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 Three Signature

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

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

Example 4. SOAP Web Services Signature

```
tIcC5zyKUmycB5Ml/cNxOHDusw03Y5KPQiXVNUHHp4U=
```

Example 5. RESTlets Signature

```
KK4SKNgz4ZiILGLwOMtfYlgcXSy1eis8ldE9X90azQ=
```

Example 6. REST Web Services Signature

```
B5OIWznZ2YP0OB7VrJrGkYsTh%2B8H%2B5T9Hag%2Bo92q0zY%3D
```

The Authorization Headers

This section covers creating authorization headers. The values used in the following code samples are defined in the section The Signature for Web Services and RESTlets.

Warning: The end of support for the HMAC-SHA1 signature method is targeted for 2023.1. You should update your integrations to use the HMAC-SHA256 signature method as soon as possible.

Create the Authorization Header

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

Note: For RESTlets, each parameter must be rawurlencoded.

See the following sections:

- SOAP Web Services Header
- RESTlet Header

SOAP Web Services Header

```
$passport = " <ns:tokenPassport soap:actor=\"http://schemas.xmlsoap.org/soap/actor/next\" soap:mustUnderstand=\"0\" xmlns:ns=\"urn:messages_2015_2.platform.webservices.netsuite.com\">\n"
  ." <ns:account>".$realm ."</ns:account>\n"
  ." <ns:consumerKey>".$consumerKey ."</ns:consumerKey>\n"
  ." <ns:token>". $tokenKey ."</ns:token>\n"
  ." <ns:nonce>". $nonce ."</ns:nonce>\n"
  ." <ns:timestamp>". $timestamp ."</ns:timestamp>\n"
  ." <ns:signature algorithm=\"". $signatureMethod ."\">". $signature ."</ns:signature>\n"
  ." </ns:tokenPassport>";
```

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>f40afdd8abaac111b13825dd5e5e2ddddb44f86d5a0dd6dcf38c20aae6b67e4</ns:consumerKey>
  <ns:token>2b0ce516420110bcbd36b69e99196d1b7f6de3c6234c5afb799b73d87569f5cc</ns:token>
  <ns:nonce>fjaLirsIcCGVZWzBX0pg</ns:nonce>
  <ns:timestamp>1508242306</ns:timestamp>
  <ns:signature algorithm="HMAC-SHA256">76wQrUWF8i3BwfAjrNnTxjFo+Ixj9YzYgsj+HVeGQyY=</ns:signature>
</ns:tokenPassport>
```

RESTlet Header

```
$header = 'Authorization: OAuth '
         .'realm="' .rawurlencode($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) .'"'
```

RESTlet Header Example

```
Authorization: OAuth realm="123456", oauth_consumer_key="ef40afdd8abaac111b13825dd5e5e2ddddb44f86d5a0dd6dcf38c20aae6b67e4",
 oauth_token="2b0ce516420110bcbd36b69e99196d1b7f6de3c6234c5afb799b73d87569f5cc", oauth_nonce="fjaLirsIcCGVZWzBX0pg", oauth_timestamp="1508242306", oauth_signature_method="HMAC-SHA256", oauth_version="1.0", oauth_signature="7mpNx1RdQn4VLSyeEwCK7jFBjGQ0blzwDSMU9Kg5Rmg%3D"
```

REST Web Services Header

```
$header = 'Authorization: OAuth '
         .'realm="' .rawurlencode($realm) .'", '
         .'oauth_token="' .rawurlencode($tokenKey) .'", '
         .'oauth_consumer_key="' .rawurlencode($consumerKey) .'", '
         .'oauth_nonce="' .rawurlencode($nonce) .'", '
         .'oauth_timestamp="' .rawurlencode($timestamp) .'", '
         .'oauth_signature_method="' .rawurlencode($signatureMethod) .'", '
```

```
8           .'oauth_version="' .rawurlencode($version) .'", '
9           .'oauth_signature="' .rawurlencode($signature) .'"'
```

REST Web Services Header Example

```
Authorization: OAuth realm="123456", oauth_token="2b0ce516420110bcbd36b69e99196d1b7f6de3c6234c5afb799b73d87569f5cc", oauth_consumer_key="ef40afdd8abaac111b13825dd5e5e2ddddb44f86d5a0dd6dcf38c20aae6b67e4", oauth_nonce="fjaLirsIcCGVZWzBX0pg", oauth_timestamp="1508242306", oauth_signature_method="HMAC-SHA256", oauth_version="1.0", oauth_signature="B5OIWznZ2YP0OB7VrJrGkYsTh%2B8H%2B5T9Hag%2Bo92q0zY%3D"
```

The RESTlet Base String

The values used in the following code samples are defined in the section The Signature for Web Services and RESTlets.

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 a screenshot 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.

```
1  function restletBaseString($httpMethod, $url, $consumerKey, $tokenKey, $nonce, $timestamp, $version, $signatureMethod, $postParams){
2    //http method must be upper case
3    $baseString = strtoupper($httpMethod) .'&';
4
```

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)

```
5    //include url without parameters, schema and hostname must be lower case
6    if (strpos($url, '?')){
7      $baseUrl = substr($url, 0, strpos($url, '?'));
8      $getParams = substr($url, strpos($url, '?') + 1);
9    } else {
10     $baseUrl = $url;
11     $getParams = "";
12   }
13   $hostname = strtolower(substr($baseUrl, 0,  strpos($baseUrl, '/', 10)));
14   $path = substr($baseUrl, strpos($baseUrl, '/', 10));
15   $baseUrl = $hostname . $path;
16   $baseString .= rawurlencode($baseUrl) .'&';
```

3. Parameters - lines 19-51

- Place 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 sorted in alphabetical order 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)

```
19    $params = array();
20    $params['oauth_consumer_key'] = array($consumerKey);
21    $params['oauth_token'] = array($tokenKey);
22    $params['oauth_nonce'] = array($nonce);
23    $params['oauth_timestamp'] = array($timestamp);
24    $params['oauth_signature_method'] = array($signatureMethod);
25    $params['oauth_version'] = array($version);
26
27    foreach (explode('&', $getParams ."&". $postParams) as $param) {
28      $parsed = explode('=', $param);
29      if ($parsed[0] != "") {
30        $value = isset($parsed[1]) ? urldecode($parsed[1]): "";
31        if (isset($params[urldecode($parsed[0])])) {
32          array_push($params[urldecode($parsed[0])], $value);
33        } else {
34          $params[urldecode($parsed[0])] = array($value);
35        }
36      }
37    }
38
39    //all parameters must be alphabetically sorted
40    ksort($params);
41
42    $paramString = "";
43    foreach ($params as $key => $valueArray){
44      //all values must be alphabetically sorted
45      sort($valueArray);
46      foreach ($valueArray as $value){
47        $paramString .= rawurlencode($key) . '='. rawurlencode($value) .'&';
48      }
49    }
50    $paramString = substr($paramString, 0, -1);
51    $baseString .= rawurlencode($paramString);
52    return $baseString;
53  }
```

The restletBaseString Function

```
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 sorted in alphabetical order , next each key and values is encoded and finally whole parame
ters 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 sorted in alphabetical order
  ksort($params);

  $paramString = "";
  foreach ($params as $key => $valueArray){
    //all values must be sorted in alphabetical order
    sort($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 for Web Services and RESTlets
- 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.

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

For information about related tasks, see the following topics:

- Create Integration Records for Applications to Use TBA
- Regenerating a Consumer Key and Secret

Note: Tokens created using the Token-based Authentication feature in your NetSuite production account are not copied to your Release Preview or to your sandbox accounts. To test this feature in Release Preview or in a sandbox, you must create new tokens in that account. Each time the sandbox is refreshed, you must create new tokens in the sandbox.

Follow the OAuth 1.0 specification to generate a token. For more information and an example, see Step One Obtain An Unauthorized Request Token.

Token-based Authentication and Web Services

SuiteTalk (web services) supports the Token-based Authentication (TBA) feature.

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.

Note: Tokens created using the Token-based Authentication feature in your NetSuite production account are not copied to your Release Preview or to your sandbox accounts. To test this feature in Release Preview or in a sandbox, you must create new tokens in that account. Each time the sandbox is refreshed, you must create new tokens in the sandbox.

For guidance on adapting an integration to include TBA credentials and to see an example that includes code samples 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:

- Requirements for Using Token-Based Authentication
- Regenerating a Consumer Key and Secret
- SOAP Web Services Governance for Token-Based Authentication
- The Three-Step TBA Authorization Flow
- RESTlet Header

Token-based Authentication and SuiteAnalytics Connect

SuiteAnalytics Connect supports token-based authentication (TBA) for the NetSuite2.com data source.

Note: Tokens created using the Token-based Authentication feature in your NetSuite production account are not copied to your Release Preview or to your sandbox accounts. To test this feature in Release Preview or in a sandbox, you must create new tokens in that account. Each time the sandbox is refreshed, you must create new tokens in the sandbox.

To access Connect with TBA, you need to create a token password which is an authentication string. For information about how to use token-based authentication with SuiteAnalytics Connect, see the help topic Token-based Authentication for Connect.

OAuth 2.0

NetSuite supports OAuth 2.0, a robust authorization framework. This authorization framework enables client applications to use a token to access NetSuite through REST web services, RESTlets, and SuiteAnalytics Connect. The application accesses the protected resources on behalf of a user who gave an explicit permission for the access. This method eliminates the need for integrations to store user credentials. OAuth 2.0 can be used as an alternative to the Token-based Authentication feature. It is more straightforward to implement, because request signing is not required.

The OAuth 2.0 feature is for use with RESTlets, REST web services, and SuiteAnalytics Connect only. It is not supported for SOAP web services.

Note: OAuth 2.0 is the preferred authentication method. You should consider using OAuth 2.0 instead of TBA whenever possible.

See the following topics for more information about OAuth 2.0:

- OAuth 2.0 Tasks for Administrators
  - Getting Started with OAuth 2.0
  - Managing OAuth 2.0 Authorized Applications
  - OAuth 2.0 Client Credentials Setup
- OAuth 2.0 for Integration Application Developers
- Troubleshooting OAuth 2.0
- OAuth 2.0 for RESTlets
- OAuth 2.0 for REST Web Services

OAuth 2.0 Tasks for Administrators

This section provides information about tasks that administrators must complete to support OAuth 2.0 feature for integrations. See the following topics:

- Getting Started with OAuth 2.0
  - Enable the OAuth 2.0 Feature
  - Set Up OAuth 2.0 Roles and OAuth 2.0 Permissions.
  - Assign Users to OAuth 2.0 Roles
  - Create Integration Records for Applications to Use OAuth 2.0
- Managing OAuth 2.0 Authorized Applications
- OAuth 2.0 Client Credentials Setup

Getting Started with OAuth 2.0

To set up OAuth 2.0 in your NetSuite account, you must complete the following tasks.

Click the links in the following steps for detailed instructions for each task.

To set up OAuth 2.0 in your NetSuite account:

1. Enable the OAuth 2.0 Feature
2. Set Up OAuth 2.0 Roles

See also OAuth 2.0 Permissions

3. Assign Users to OAuth 2.0 Roles
4. Create Integration Records for Applications to Use OAuth 2.0

See also Managing OAuth 2.0 Authorized Applications, and OAuth 2.0 Client Credentials Setup

Enable the OAuth 2.0 Feature

Before you can begin using OAuth 2.0 in your account, you must enable the feature.

To enable OAuth 2.0 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: You must enable both the Client SuiteScript and Server SuiteScript features to use OAuth 2.0 feature for RESTlets.
4. In the **Manage Authentication** section, check the **OAuth 2.0** box. Click **I Agree** on the SuiteCloud Terms of Service page.
5. Click **Save**.

   Note: The **Manage OAuth 2.0 Authorized Applications** link becomes available in the Settings portlet for users with a role that has been assigned Log in Using OAuth 2.0 Access Token permission. Users can only list their own OAuth 2.0 authorized applications through this link. Administrators and users with OAuth 2.0 Authorized Applications Management permission can list all authorized applications in the account on Setup > Users/Roles > OAuth 2.0 Authorized Applications.

After you have enabled the OAuth 2.0 feature:

- You must set up OAuth 2.0 roles. See Set Up OAuth 2.0 Roles. See also OAuth 2.0 Permissions.
- Administrators and users with the Integration Application permission can configure applications to use OAuth 2.0 to access RESTlets, REST web services, and SuiteAnalytics Connect. See Create Integration Records for Applications to Use OAuth 2.0.

Set Up OAuth 2.0 Roles

An administrator can create a new role with OAuth 2.0 permissions, or modify existing roles to add OAuth 2.0 permissions, then assign users to these roles as needed. If you need more information about creating or customizing roles, see:

- NetSuite Users & Roles
- NetSuite Roles Overview

OAuth 2.0 Permissions

The following OAuth 2.0 permissions can be added to roles as appropriate.

- **OAuth 2.0 Authorized Applications Management**:

- Is primarily for the Administrator role, or roles with Core Administration Permissions (CAP). For more information about CAP, see the help topic Core Administration Permissions.
- Requires two-factor authentication (2FA).
- Enables users to view or revoke any OAuth 2.0 authorized applications in the account. For more information, see Managing OAuth 2.0 Authorized Applications
- Enables users to set up the OAuth 2.0 client credentials flow in the account. For more information, see OAuth 2.0 Client Credentials Setup.

- **Log in using OAuth 2.0 Access Tokens** – enables users to:
  - Access REST web services, RESTlets, and SuiteAnalytics Connect using OAuth 2.0 access tokens.
  - View their OAuth 2.0 authorized applications. For more information, see Managing OAuth 2.0 Authorized Applications
  - Revoke OAuth 2.0 authorized applications they authorized previously.

To add permissions to a role, go to Setup > Users/Roles > Manage Roles. Select a role to customize. On the Permission tab, Setup subtab, choose the permission from the list and click **Add**.

Note: A user assigned the OAuth 2.0 Authorized Applications Management permission cannot access RESTlets, REST web services, and SuiteAnalytics Connect using OAuth 2.0. To use OAuth 2.0 for access to RESTlets, REST web services, and SuiteAnalytics Connect, the user must be assigned a role that has the Log in Using OAuth 2.0 Access Tokens permission.

For more information, see Assign Users to OAuth 2.0 Roles.

Assign Users to OAuth 2.0 Roles

After you have modified roles to add OAuth 2.0 permissions, you can assign users to these roles. OAuth 2.0 is available for many types of NetSuite users, including customers, employees, partners, contacts and vendors. The following is a brief procedure for assigning a role to an existing user. If you need more information about assigning roles to users, see the help topic NetSuite Users Overview.

To assign OAuth 2.0 roles to users:

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 click **Customers**, **Partners**, or **Vendors**.
2. Click **Edit** next to the name of the user to whom you want to assign the OAuth 2.0 role.
3. Click the **Access** tab.
4. On the Roles subtab, in the **Role** field, select the OAuth 2.0 role for this user.
5. Click **Add**.
6. Click **Save**.

Next, you must set up applications to use OAuth 2.0 for authentication. See Create Integration Records for Applications to Use OAuth 2.0

Create Integration Records for Applications to Use OAuth 2.0

Before users can authorize an OAuth 2.0 application for NetSuite access through REST web services, RESTlets, or SuiteAnalytics Connect, an integration record must be created for the application. It is also possible to edit an existing integration record. Administrators or users with the Integration Application

permission can create integration records. For more information about integration records, see the help topic Integration Management.

The following procedure describes how to create an integration record.

To create an integration record for an application:

1. Go to Setup > Integration > New.
2. Enter a name for your application in the **Name** field.
3. Enter a description in the **Description** field, if preferred.
4. Select Enabled in the **State** field.
5. Enter a note in the **Note** field, if preferred.

> **Note:** Values of the **State** , **Note**, and **OAuth 2.0 Consent Policy** fields are specific to one NetSuite account. If you install a record in a different account, the values may change. Values of the **Name** and **Description** fields are read-only if the record is installed in a different account. For more information, see the help topic Auto-Installation of Integration Records.

6. On the **Authentication** tab, check the appropriate boxes for your application:

Field on the Authentication tab, under OAuth 2.0:	Function of the field:
Authorization Code Grant For more information, see OAuth 2.0 for Integration Application Developers.	Check this box if you want to implement the OAuth 2.0 authorization code grant flow for this integration. **Note:** You can check both the **Authorization Code Grant** box and the **Client Credentials (Machine to Machine Grant)** box.
Redirect URI	▪ Enter the valid redirect URI for your application, on which the authorization code will be handled. ▪ The redirect URI is validated when you save the integration record. **Important:** The redirect URI must be configured as either the https:// scheme or a custom URL scheme (for example, myapp:// callback). The http:// scheme is not supported. The transport layer security must be guaranteed on the redirect URI.
Public Client	(Optional). Check this box if you want to allow OAuth 2.0 public clients with this integration. **Important:** The Client Credentials (Machine to Machine) Grant does not support the use of public clients.
Client Credentials (Machine to Machine) Grant For more information, see OAuth 2.0 for Integration Application Developers.	Check this box if you want to implement the OAuth 2.0 client credentials flow for this integration. **Note:** You can check both the **Authorization Code Grant** box and the **Client Credentials (Machine to Machine Grant)** box.
RESTlets For more information, see OAuth 2.0 for RESTlets.	Check this box if your OAuth 2.0 integration application requires accessing RESTlets.

REST Web Services For more information, see OAuth 2.0 for REST Web Services.	Check this box if your OAuth 2.0 integration application requires accessing REST web services.
SuiteAnalytics Connect	Check this box if your OAuth 2.0 integration application requires accessing SuiteAnalytics Connect.
Application Logo	(Optional). You can select a file from your File Cabinet. Supported formats are JPEG, PNG and GIF.
Application Terms of Use	(Optional). You can select any PDF file from your File Cabinet.
Application Privacy Policy	(Optional). You can select any PDF file from your File Cabinet.
OAuth 2.0 Consent Policy	Select an option from the list. See the following for more details about these options: ■ **Always Ask** - This is the default option. The consent screen appears every time the OAuth 2.0 code grant flow is initiated. ■ **Never Ask** - The consent screen does not appear during the OAuth 2.0 code grant flow. The integration is autoapproved by the Administrator. ■ **Ask First Time** - The consent screen only appears the first time the OAuth 2.0 code grant flow is initiated. The consent screen also appears if: □ The consent was not given previously □ The system does not know which role or account to choose for the user to log in with □ The application requires a different set of scopes and needs a new consent Integration application developers can adjust the consent screen option using the prompt parameter in Step One of the OAuth 2.0 code grant flow. For more information, see Step One GET Request to the Authorization Endpoint. See also Integration Record and Prompt Parameter Combinations.

Note: If you do not want use the Token-based Authentication feature for your integration, clear the **TBA: Authorization Flow** box, then clear the **Token-based Authentication**.

7. Click **Save**.

Warning: For security reasons, the only time the Client Credentials (the client ID and client 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 the client ID and client secret, you will have to reset them on the Integration page, to obtain new values. Treat these values as you would a password. Never share the client ID and client secret with unauthorized individuals and never send them by email.

Enabling an Application to Use OAuth 2.0

In some cases, you may have an existing application that is not set up for OAuth 2.0. For example, you may have configured an application to authenticate through user credentials, which is not a safe authentication method, and you should not use it.

To enable OAuth 2.0 for an existing application:

1. Go to Setup > Integration > Managing Integrations, and open the appropriate integration record for editing.

Important: OAuth 2.0 is only available for RESTlets and REST web services.

2. Check the **Authorization Code Grant** box, the **Client Credentials (Machine to Machine) Grant**, or both.
3. Chose a scope and check the appropriate box, **RESTlets**, **REST Web Services**, **SuiteAnalytics Connect**, or all of them.
4. Define the **Redirect URI**.
5. (Optional). Choose the **Application Logo**, **Application Terms of Use**, and **Application Privacy Policy**.
6. (Optional). Check the **Public Client** box, if you want to allow OAuth 2.0 public clients with this integration.
7. Select an option from the **OAuth 2.0 Consent Policy** list.
8. Click **Save**.

Important: The system displays the client ID and client secret only the first time you save the integration record. In cases where an application previously used user credentials as an authentication method, you must reset the client ID and client secret. In cases where the application used TBA, the client ID and client secret are not displayed. You can either use the same values as you used for TBA or reset them to get new values.

Warning: Resetting the client ID and client secret invalidates the previous client ID and client secret. This may invalidate the access token previously used as authentication method of the integration record.

Managing OAuth 2.0 Authorized Applications

Note: Applications authorized using the OAuth 2.0 feature in your NetSuite production account are not copied to your Release Preview or to your sandbox accounts. Users must authorize applications explicitly in Release Preview or in a sandbox to test OAuth 2.0 feature in these accounts. Each time the sandbox is refreshed, users must authorize applications explicitly in the sandbox.

Management of authorized applications includes the following tasks:

- Viewing OAuth 2.0 Authorized Applications
- Revoking OAuth 2.0 Authorized Applications

Important: Unlike the TBA tokens, you cannot create OAuth 2.0 authorized applications directly in the UI. For more information, see OAuth 2.0 for Integration Application Developers.

Viewing and Revoking OAuth 2.0 Authorized Applications

You can see a list view of authorized applications in your system. Users in roles with OAuth 2.0 Authorized Applications Management permission can also see a list view of all authorized applications in the account.

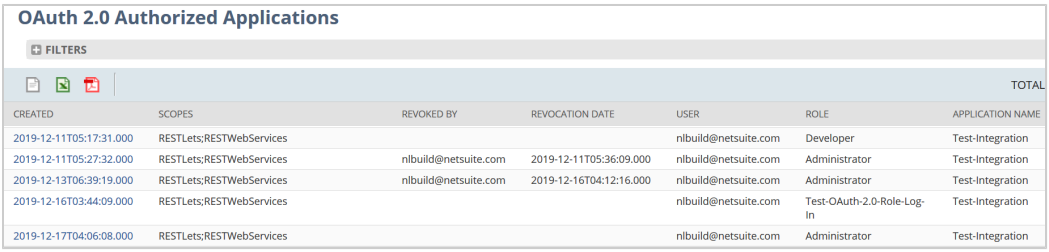
Viewing OAuth 2.0 Authorized Applications

See the following procedure for detailed information about how to view the OAuth 2.0 authorized applications.

To view authorized applications

1. Go to Setup > Users/Roles > OAuth 2.0 Authorized Applications.

   The OAuth 2.0 Authorized Applications page appears.



   Users with Log in using OAuth 2.0 Access Tokens permission can only access this page directly from the Settings portlet by clicking the **Manage OAuth 2.0 Authorized Applications** link.

   Important: Users can only list their own OAuth 2.0 authorized applications through the Settings portlet link.

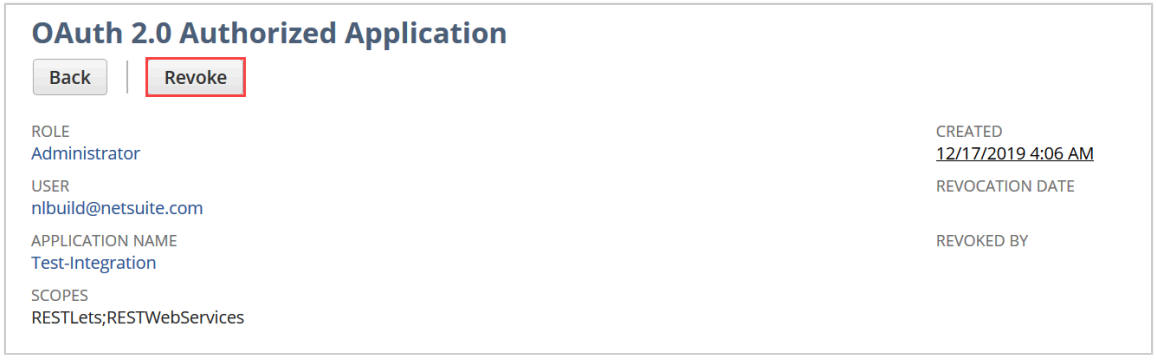
2. To view the authorized application, click the link in **Created** column.

Revoking OAuth 2.0 Authorized Applications

Revoking an authorized application invalidates all tokens associated with that application. However, the application record is still present in the system and is still accessible for auditing purposes.

Note: The user who authorized the revoked application previously, must give a new consent on a consent screen. This action creates a new authorized application record, with a new pair of tokens.

To revoke a permission for an authorized application, go to OAuth 2.0 Authorized Applications page, click the link in **Created** column and click **Revoke**. After this action, the system enters values in the **Revocation Date** field and the **Revoked By** field.



Note: In case the scope of protected resources (RESTlets or REST web services) is extended in the integration record, the system revokes the authorized application for that record and automatically creates a new one.

OAuth 2.0 Client Credentials Setup

Note: The client credentials flow setup in your NetSuite production account is not copied to any other production account, Release Preview account, or sandbox account. Users must set up the flow explicitly in each account, including Release Preview accounts, and sandbox accounts, to test the OAuth 2.0 client credentials flow in these accounts. Each time a sandbox account is refreshed, users must explicitly set up the flow in that sandbox account.

An administrator, or a user logged in with a role that has the OAuth 2.0 Authorized Applications Management permission, can create or revoke a mapping for the OAuth 2.0 client credentials flow.

Creation of this mapping is a required step for the OAuth 2.0 client credentials flow to work.

Creating a Mapping for the Client Credentials Flow

See the following procedure for steps to create a new mapping for the OAuth 2.0 client credentials flow.

To create a mapping for the client credentials flow:

1. Go to Setup > Integration > Manage Authentication > OAuth 2.0 Client Credentials (M2M) Setup.

   The OAuth 2.0 Client Credentials Setup page appears.
2. To create a new mapping, click the **Create New** button.
3. In the popup window, choose the entity, role, and application to be mapped. Upload the public part of the certificate from your computer.

   Note: In the popup window, the application is only available to be selected if the **Client Credentials (Machine to Machine) Grant** box is checked on the associated integration record. For more information, see Create Integration Records for Applications to Use OAuth 2.0.
4. Click **Save**.

The mapping is added to the list of all created mappings on the OAuth 2.0 Client Credentials (M2M) Setup page. The list includes the data you entered, as well as the data imported from the certificate. There is a record for every unique combination of application and certificate.

To revoke a certificate, click the **Revoke** button in the Revoked column.

If you revoke a certificate or the certificate expires, you must create a new mapping for the integration record to continue using the OAuth 2.0 client credentials flow.

Certificate Conditions

A certificate to be used with the OAuth 2.0 client credentials flow must contain two parts. An Administrator or a user with the OAuth 2.0 Authorized Applications Management permission uploads the public part of the certificate as part of the client credentials flow mapping process. The Administrator or a user with the OAuth 2.0 Authorized Applications Management permission does not upload the private part of the certificate. The private part of the certificate provides the signature of the JWT token in the

POST request to the token endpoint. For more information, see POST Request to the Token Endpoint and the Access Token Response.

The certificate must meet the following requirements:

- The public part of the certificate must be in x.509 format with a file extension of .cer, .pem, or .crt.
- The length of the RSA key must be 3072 bits, or 4096 bits. The length of EC key must 256 bits, 384 bits, or 521 bits.
- The maximum time period that a certificate can be valid is two years. If the certificate is valid for a longer time period, the system automatically shortens the time period to two years.
- One certificate can only be used for one combination of integration record, role, and entity. If you want to use the same integration record for multiple entities or roles, you must use a different certificate for each unique combination.

The following example shows how to create a valid certificate using OpenSSL:

openssl req -x509 -newkey rsa:4096 -sha256 -keyout auth-key.pem -out auth-cert.pem -nodes -days 730.

Important: Treat the certificate as you would any other credentials. Never share the certificate with unauthorized individuals, or outside your company.

OAuth 2.0 for Integration Application Developers

OAuth 2.0 access is based on the authorization code grant flow for the generation of access tokens and refresh tokens, or the client credentials flow. The client credentials flow is a machine-to-machine flow for the generation of access tokens.

These alternatives are more straightforward than the three-step TBA authorization flow, because they do not require signing of requests.

For more information about the OAuth 2.0 code grant flow, see OAuth 2.0 Authorization Code Grant Flow.

For more information about the OAuth 2.0 client credentials flow, see OAuth 2.0 Client Credentials Flow.

OAuth 2.0 Authorization Code Grant Flow

Application developers and integrators can use a redirection-based authorization code grant flow with OAuth 2.0. If there is no active session, users enter user credentials into one of the following login forms as a part of the flow.

- A trusted NetSuite login form
- SAML SSO Identity provider's login form
- OIDC OpenID Connect provider's login form

The OAuth 2.0 authorization code grant flow consists of two steps, an additional refresh token request, and a request to the revoke token endpoint.

- Step One GET Request to the Authorization Endpoint
- Step Two POST Request to the Token Endpoint
- Refresh Token POST Request to the Token Endpoint
- POST Request to the Revoke Token Endpoint

With the OAuth 2.0 authorization code grant flow, the application begins the process of granting the access token and refresh token by sending a GET request to the authorization endpoint. The user,

to whom the access token and refresh token are to be granted, explicitly consents to the application accessing NetSuite through RESTlets, REST web services, or SuiteAnalytics Connect.

The Administrator must create an integration record for each application. See Create Integration Records for Applications to Use OAuth 2.0. The Administrator must configure the redirect URI on the integration record. The underlying application must have the ability to open a browser.

For more information, see RFC 6749.

Step One GET Request to the Authorization Endpoint

In the first step of the OAuth 2.0 authorization code grant flow, the application sends a GET request to the authorization endpoint. This request must include the required parameters in the request header.

The format of the URL is:

`https://<accountID>.app.netsuite.com/app/login/oauth2/authorize.nl`.

where <accountID> represents your NetSuite account ID. If you do not know the specific account ID, requests can be sent to

`https://system.netsuite.com/app/login/oauth2/authorize.nl`.

See the following table for details about parameters for the GET request.

Request Parameters for Step One

Request Parameter	Description
response_type	The value of the response_type parameter is always code.
client_id	■ Identifies the client. ■ The value of the client ID is provided when the integration record is created.
redirect_uri	■ The application uses the valid redirect URI to handle the authorization code. ■ The value of the redirect URI parameter must match the redirect URI in the corresponding integration record.
scope	■ The scope for which the application is requesting access. Values are `restlets`, `rest_webservices`, `suite_analytics`, or all of them. If the application requests access for both, the values are separated by a white space. ■ The requested scope must be enabled in the corresponding integration record. For more information, see Create Integration Records for Applications to Use OAuth 2.0. **Note:** NetSuite supports two additional scopes that are used for the NetSuite as OIDC Provider feature. For more information, see Step One GET Request to the Authorization Endpoint in the NetSuite as OIDC Provider topic.
state	The length of the state parameter must be between 24 and 1024 characters. Valid characters are all printable ASCII characters. **Important:** To avoid cross-site request forgery (CSRF) attacks, you must conform to the OAuth 2.0 specification. For more information, see RFC6749 Section 10.12.
code_challenge	■ This parameter is optional; however, you should use this security extension.

Request Parameter	Description
	Important: If you use public clients for OAuth 2.0, the code_challenge parameter is required. ■ The code_challenge parameter is created using code_verifier, a random string of characters. For more information, see PKCE specification https://tools.ietf.org/html/rfc7636#section-4.2. ■ Apply SHA256 on the code_verifier parameter. ■ The length of the code_verifier parameter must be between 43 and 128 characters. ■ Valid characters for the code_verifier parameter are alphabet characters, numbers, and non-alpha numeric ASCII characters: hyphen, period, underscore, and tilde (- . _ ~).
code_challenge_method	■ This parameter is optional. However, if you configure the code_challenge parameter, you must configure the code_challenge_method parameter. You must use SHA256 to configure the code_challenge_method parameter. ■ When the authorization server generates an authorization code, the code_challenge and code_challenge_method parameters are associated with the authorization code value, to ensure they are properly verified. For more information, see PKCE specification https://tools.ietf.org/html/rfc7636#section-4.4. **Important:** As of 2020.2, NetSuite does not support a value of `plain` for the code_challenge_method parameter. Use S256 instead.
prompt	The optional prompt parameter provides additional control of when the consent screen appears. Following are the values you can use with the prompt parameter: ■ `none` - the consent screen does not appear. If there is no active session, the application returns an error. ■ `login` - the user must authenticate even if there is an active session. This option only works if the application sends the request to the account-specific domain. ■ `consent` - the consent screen appears every time. The user must authenticate if there is no active session. ■ `login consent` or `consent login` - the consent screen appears every time, and the user must authenticate even if there is an active session. For more information, see Integration Record and Prompt Parameter Combinations.

Important: Request parameters must be encoded based on the HTML specification for the `application/x-www-form-urlencoded` media type. For more information, see URL Specification 5.1.

The following URL provides a sample GET request.

```
https://<accountID>.app.netsuite.com/app/login/oauth2/authorize.nl?scope=restlets+rest_webservices&redirect_uri=https%3A%2F%2Fmyapplication.com%2Fnetsuite%2Foauth2callback&response_type=code&client_id=6794a3086e4f61a120350d01b8527aed3631472ef33412212495be65a8fc8d4c&state=ykv2XLx1BpT5Q0F3MRPHb94j&code_challenge=Who5QBshz2Mu1Mq6GuAknYA5TnjA-0z7VhAgLloec1s&code_challenge_method=S256
```

Consent Screen

After the application sends the GET request, the system displays the consent screen, where a user authorizes the application to access NetSuite through RESTlets, REST web services, or SuiteAnalytics Connect.

Important: If there is no active NetSuite session, the user is first redirected to the NetSuite login form. If the GET request points to an account-specific domain, for an account with SAML SSO or OIDC enabled, the user may be redirected to a third party application. After successful authentication, the system displays the consent screen.

The consent screen includes the following:

- The **Application Logo**, **Terms of Use**, and **Privacy Policy**, if these values were entered in the integration record.
- A data and a role to which the application requests access.
- The **Allow/Continue** button. If the application was not previously authorized, the user must click **Allow** to authorize the application. If the application was previously authorized, the user must click a **Continue** button to continue to the next step of the flow.
- The **Deny/Go Back** button. If the application was not previously authorized, the user can click **Deny** to interrupt the flow. If the application was previously authorized, the user can click **Go Back** to interrupt the flow.
- The **Choose Another Role** list. The user can change a role that authorizes the application, by clicking the **Choose Another Role** link.

Redirect Parameters for Step One

After authorization, NetSuite initiates a redirect to the Redirect URI, with the following parameters:

Redirect Parameter	Description
state	The state parameter in the redirect must match the state parameter in the request in Step One. To avoid cross-site request forgery (CSRF) attacks, you must conform to the OAuth 2.0 specification. For more information, see RFC6749 Section 10.12.
code	▪ A randomly generated string that is used for request verification in Step Two. ▪ The code parameter is only generated if the application was authorized. ▪ You must use the value of the code parameter immediately after it is generated. The value for the code parameter has limited time validity.
role	Indicates the user's role for which the access token and refresh token are granted in Step Two. The role parameter is a NetSuite-specific parameter.
entity	The ID of the user who authorizes the application or interrupts the flow. The entity parameter is a NetSuite-specific parameter.
company	NetSuite account ID (company identifier). The company parameter is a NetSuite-specific parameter.
error	The error parameter is only used when an error occurs during the flow. For information about error values, see Troubleshooting OAuth 2.0.

The following sample redirects illustrate successful and unsuccessful authorization.

- Application successfully authorized:

```
https://myapplication.com/netsuite/oauth2callback?state=ykv2XLx1BpT5Q0F3MRPHb94j&role=1000&entity=12&company=1234567&code=70b827f926a512f098b1289f0991abe3c767947a43498c2e2f80ed5aef6a5c50
```

- Application not authorized:

```
https://myapplication.com/netsuite/oauth2callback?state=ykv2XLx1BpT5Q0F3MRPHb94j&role=1000&entity=12&company=1234567&error=access_denied
```

After the request to the redirect URI is sent, the flow proceeds to Step Two POST Request to the Token Endpoint.

Step Two POST Request to the Token Endpoint

The application sends a POST request to the token endpoint. The request must include client credentials in the HTTP authorization request header and the required parameters in the request body. At the end of this step, the access token and refresh token are granted.

Note: If you use public clients you can choose from the following options:

- The HTTP authorization request header does not contain the Authorization. Additionally, the PKCE parameters and the client_id parameter are included in the body of the request, or
- The HTTP authorization request header contains only the client_id in the Authorization. The PKCE parameters are included in the body of the request.

The format of the URL is:

`https://<accountID>.suitetalk.api.netsuite.com/services/rest/auth/oauth2/v1/token`

where <accountID> is your NetSuite account ID.

Request Parameters for Step Two

Request Parameter	Description
code	The code parameter value obtained in Step One.
redirect_uri	The value of the redirect_uri parameter must match the value entered in the corresponding integration record and the value in the request in Step One.
grant_type	The value of the grant_type parameter in Step Two is `authorization_code`.
code_verifier	If the value of the code_verifier parameter does not match the value generated in Step One, an HTTP 400 Bad Response error is returned. For more information, see https://tools.ietf.org/html/rfc7636, sections 4.5 and 4.6.

Important: Be aware of the following requirements for the request:

- Request parameters must be encoded based on the HTML specification for the `application/x-www-form-urlencoded` media type. For more information, see URL Specification 5.1
- The client authentication method used in the header of the request follows the HTTP Basic authentication scheme. For more information, see RFC 7617. The format is **clientid:clientsecret**. The string value is Base64 encoded. The following code provides an example.

```
POST /services/rest/auth/oauth2/v1/token HTTP/1.1
```

```
Host: <accountID>.suitetalk.api.netsuite
Authorization: Basic Njc5NGEzMDg2ZTRmNjFhMTIwMzUwZDAxYjg1MjdhZWQzNjMxNDcyZWYzMzQxMjIxMjQ5NWJlNjVhOGZjOGQ0YzpjZGM3YWMyMjE4M2VmNTAyNGU4MWIwZmNlOGVmNDYxYzQ0ZDU4OTZhMWYxODA1ZDRiMzcyY2E2MWM0ZDMyNmFl
Content-Type: application/x-www-form-urlencoded

code=70b827f926a512f098b1289f0991abe3c767947a43498c2e2f80ed5aef6a5c50&redirect_uri=https%3A%2F%2Fmyapplication.com%2Fnetsuite%2Foauth2callback&grant_type=authorization_code&code_verifier=abFOm_isZAwm7PpI9BtJRMEuiMqhU6sUqZlVWSsAAf1QutgIOD~on78mu-JdpbKc_RA7IEcf2e~q0XrKlJ1tE.8Un64PXLKQG16G41wW-a5de_0aeU2mHnyVPg.Or8cE
```

Note: If you use public clients you can choose from the following options:

- The HTTP authorization request header does not contain the Authorization. Additionally, the PKCE parameters and the client_id parameter are included in the body of the request, or
- The HTTP authorization request header contains only the client_id in the Authorization. The PKCE parameters are included in the body of the request.

The following code provides an example of the HTTP authorization request with the PKCE parameters and the `client_id` parameter included in the body of the request:

```
POST /services/rest/auth/oauth2/v1/token HTTP/1.1
Host: <accountID>.suitetalk.api.netsuite
Content-Type: application/x-www-form-urlencoded

code=70b827f926a512f098b1289f0991abe3c767947a43498c2e2f80ed5aef6a5c50&redirect_uri=https%3A%2F%2Fmyapplication.com%2Fnetsuite%2Foauth2callback&grant_type=authorization_code%client_id=6794a3086e4f61a120350d01b8527aed3631472ef33412212495be65a8fc8d4c&code_verifier=XG2JcZ.I5_67es~Pev0ZbWSPlOklZJq-KPNaFInKuzgGQV4AWt5taWLDnD4IAsnJW_hl9iPQdQcv9~xGSY.qTiB99HA2rfm8cUwlfrzBY0j3bK4XPx-gLhoV1MF1JCC2
```

The following code provides an example of the HTTP authorization request with the `client_id` parameter included in the Authorization. The PKCE parameters are included in the body of the request:

```
POST /services/rest/auth/oauth2/v1/token HTTP/1.1
Host: <accountID>.suitetalk.api.netsuite
Authorization: Basic Njc5NGEzMDg2ZTRmNjFhMTIwMzUwZDAxYjg1MjdhZWQzNjMxNDcyZWYzMzQxMjIxMjQ5NWJlNjVhOGZjOGQ0Yzo=
Content-Type: application/x-www-form-urlencoded

code=70b827f926a512f098b1289f0991abe3c767947a43498c2e2f80ed5aef6a5c50&redirect_uri=https%3A%2F%2Fmyapplication.com%2Fnetsuite%2Foauth2callback&grant_type=authorization_code%code_verifier=XG2JcZ.I5_67es~Pev0ZbWSPlOklZJq-KPNaFInKuzgGQV4AWt5taWLDnD4IAsnJW_hl9iPQdQcv9~xGSY.qTiB99HA2rfm8cUwlfrzBY0j3bK4XPx-gLhoV1MF1JCC2
```

HTTP Response for Step Two

JSON Response Fields	Description
access_token	The value of the access_token parameter is in JSON Web Token (JWT) format. The access token is valid for 60 minutes.
refresh_token	The value of the refresh_token parameter is in JSON JWT format. The refresh token is valid for seven days. **Important:** If you use public clients for OAuth 2.0, the refresh token is only valid for three hours and is for one-time use only.
expires_in	The value of the expires_in parameter is always `3600`. The value represents the time period during which the access token is valid, in seconds.

JSON Response Fields	Description
token_type	The value of the token_type parameter is always `bearer`.
id_token	This parameter is a part of OAuth 2.0, but it is used only in the NetSuite as OIDC Provider feature flow. You do not need to configure the token_id parameter as a part of the OAuth 2.0 feature flow. For more information, see Step Two POST Request to the Token Endpoint.

The following is an example of a response in JSON JWT format:


```
{"access_token":"eyJraWQiOiJzLlNZU1RFTS4yMDIwXzEiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiIzOzciLCJhdWQiOlsiNkFDQkQ1MUMt
NTE0Qi00RjU5LUIxQzMtQ0IzNUZGN0U5QTZBOzQwMzAwNTkiLCI0Nzc2MWI3NzY1MjJlMTg2ZmNkNzdmMTNjMjVlOGNjZjk5YWY5MDFhNjc4YTY2ZTcwMGIxNjFlZW
ZlOGZhODhkIl0sInNjb3BlIjpbInJlc3Rfd2Vic2VydmljZXMiLCJyZXN0bGV0cyJdLCJpc3MiOiJodHRwczpcL1wvc3lzdGVtLm5ldHN1aXRlLmNvbSIsIm9pdCI6M
TYxMTA2NzY1NSwiZXhwIjoxNjExMDcxMjU1LCJpYXQiOjE2MTEwNjc2NTUsImp0aSI6IjQwMzAwNTkuYS41YjMyMzZiOS1mZmVlLTQyZDMtYmQ1Ny00YmU3YjQ0Mzlh
MzdfMTYxMTA2NzY1NTM1OS4xNjExMDY3NjU1MzU5In0.TVpquJSRujxyZpp9ydnkfQFy8fq2eTRIt-7mA6B9nGvftEQ2pJCu-15qfxYoe6iKU1JEpOhuvA-MAzdI-
TvM1ndHT37VRdpcEa3R_kdZuDIT5hAS0G5VRVOQVF6bseHTKm4HIe0bFy8vCIaS6utQ46crF0LnQK_bxYXsQz8nFEwGlk4mOmsKje5ZB_0vzXpHEuYh9sBFdwxhMN
UO3P_tFiAF0f0oXXJzAzYTEjA9pH_tr1ymGFoLWCIfKiR1RUavvVVGeL-jiQdZSRNr5cQj4Nz8iixn9bR2R1rEtcoXBzAJ2pSVU9yimLe2bPmzxBggJr839PDUP4IlK
wkvzMUoLw","refresh_token":"eyJraWQiOiJzLlNZU1RFTS4yMDIwXzEiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiIzOzciLCJhdWQi
OlsiNkFDQkQ1MUMtNTE0Qi00RjU5LUIxQzMtQ0IzNUZGN0U5QTZBOzQwMzAwNTkiLCI0Nzc2MWI3NzY1MjJlMTg2ZmNkNzdmMTNjMjVlOGNjZjk5YWY5MDFhNjc4Y
TY2ZTcwMGIxNjFlZWZlOGZhODhkIl0sInNjb3BlIjpbInJlc3Rfd2Vic2VydmljZXMiLCJyZXN0bGV0cyJdLCJpc3MiOiJodHRwczpcL1wvc3lzdGVtLm5ldHN1aXR
lLmNvbSIsIm9pdCI6MTYxMTA2NzY1NSwiZXhwIjoxNjExNjcyNDU1LCJpYXQiOjE2MTEwNjc2NTUsImp0aSI6IjQwMzAwNTkuci41YjMyMzZiOS1mZmVlLTQyZDM
tYmQ1Ny00YmU3YjQ0MzlhMzdfMTYxMTA2NzY1NTM1OS4wIn0.BbSQ86Phg6CKLMJ9gJQurs1NK6niSxFzF2EBFT--KFysI2AV9S1llZgxsRNIrMXsDioaepdWsGzrKepJX
q25t5Sr7f-jBwTLK9g9SkFAvvEFsVJCYbdA4_BNZkHKlCC-1mA_yFNZWBYPdfCMGDX39iIDd7LVkaj-oPjpnuRnnK1ntNzxHx0coJiXwj3KfOI0PK7xfG1zbVSW14XO
latWbi80MY0ZQCgF41nFs-Rv-a7r-b51mMrm6kKZx-0MXfKRYT60H3gPXCk2QzkKovKy3kBVjajbtVPNS2tF_SbFNWOXJrn4MFzvnnDy0qsxT_Ijy3S5LTgk4YLrlwKv_X
oE7A","expires_in":"3600","token_type":"bearer"}
```


Note: The access token and refresh token are Base64 encoded. For more information, see RFC 7519.

After the access token and refresh token are granted, the integration record is auto-installed in the account. If the integration record fails to auto-install, check the **State** field in the corresponding integration record. Go to Setup > Integration > Manage Integrations, and click the name of the corresponding integration record. The value of the **State** field must be **Enabled** to auto-install the integration record successfully.

Note: The integration record should be auto-installed when the access token is granted, if the **Require Approval During Auto-installation of Integration** box is not checked on the SOAP Web Services Preferences page. If this box is checked, you must clear it to successfully auto-install integrations.

You must change the state manually if an access token was granted for the integration record before the box was cleared. To change the state manually, go to Setup > Integration > Manage Integrations, click the name of the corresponding integration record, and change the **State** field value to **Enabled**.

Refresh Token POST Request to the Token Endpoint

When the access token expires, the application can send the refresh token POST request to the token endpoint to get a new access token.

The format of the URL is:

`https://<accountID>.suitetalk.api.netsuite.com/services/rest/auth/oauth2/v1/token`

where <accountID> represents your NetSuite account ID.

Request Parameters for the Refresh Token Request

Request Parameter	Description
grant_type	The value of the grant_type parameter is `refresh_token`.
refresh_token	The value of the refresh_token parameter is in JSON Web Token (JWT) format.

Important: the client authentication method used in the header of the request follows the HTTP Basic authentication scheme. For more information, see RFC 7617. The format is **client_id:client_secret**. The string value is Base64 encoded. The following code provides an example.

```
POST /services/rest/auth/oauth2/v1/token HTTP/1.1
Host: <accountID>.suitetalk.api.netsuite.com
Authorization: Basic Njc5NGEzMDg2ZTRmNjFhMTIwMzUwZDAxYjg1MjdhZWQzNjMxNDcyZWYzMzQxMjIxMjQ5NWJlNjVhOGZjOGQ0YzpjZGM3YWMyMjE4M2VmNTAyNGU4MWIwZmNlOGVmNDYxYzQ0ZDU4OTZhMWYxODA1ZDRiMzcyY2E2MWM0ZDMyNmFl
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token&refresh_token=eyJraWQiOiJzLlNZU1RFTS4yMDIwXzEiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiIzOzciLCJhdWQiOlsiNkFDQkQ1MUMtNTE0Qi00RjU5LUIxQzMtQ0IzNUZGN0U5QTZBOzQwMzAwNTkiLCI0Nzc2MWI3NzY1MjJlMTg2ZmNkNzdmMTNjMjVlOGNjZjk5YWY5MDFhNjc4YTY2ZTcwMGIxNjFlZWZlOGZhODhkIl0sInNjb3BlIjpbInJlc3Rfd2Vic2VydmljZXMiLCJyZXN0bGV0cyJdLCJpc3MiOiJodHRwczpcL1wvc3lzdGVtLm5ldHN1aXRlLmNvbSIsIm9pdCI6MTYxMTA2NzY1NSwiZXhwIjoxNjExNjcyNDU1LCJpYXQiOjE2MTEwNjc2NTUsImp0aSI6IjQwMzAwNTkuci41YjMyMzZiOS1mZmVlLTQyZDMtYmQ1Ny00YmU3YjQ0MzlhMzdfMTYxMTA2NzY1NTM1OS4wIn0.BbSQ86Phg6CKLMJ9gJQurs1NK6niSxFzF2EBFT--KFysI2AV9S1llZgxsRNIrMXsDioaepdWsGzrKepJXq25t5Sr7f-jBwTLK9g9SkFAvvEFsVJCYbdA4_BNZkHKlCC-1mA_yFNZWBYPdfCMGDX39iIDd7LVkaj-oPjpnuRnnK1ntNzxHx0coJiXwj3KfOI0PK7xfG1zbVSW14XOlatWbi80MY0ZQCgF41nFs-Rv-a7r-b51mMrm6kKZx-0MXfKRYT60H3gPXCk2QzkKovKy3kBVjajbtVPNS2tF_SbFNWOXJrn4MFzvnnDy0qsxT_Ijy3S5LTgk4YLrlwKv_XoE7A
```

Note: If you use public clients you can choose from the following options:

- The HTTP authorization request header does not contain the Authorization, and the client_id parameter is included in the body of the request, or
- The HTTP authorization request header contains only the client_id in the Authorization.

HTTP Response for Refresh Token Request

JSON Response Fields	Description
access_token	The value of the access_token parameter is in JSON Web Token (JWT) format. The access token is valid for 60 minutes.
expires_in	The value of expires_in parameter is always `3600`. The value represents the time period during which the access token is valid, in seconds.
token_type	The value of the token_type parameter is always `bearer`.

Important: If you use public clients with OAuth 2.0, the refresh token request returns an access and refresh token. The refresh token is valid for three hours and is for one-time use only.

The following is an example of a response:

```
{"access_token":"eyJraWQiOiJzLlNZU1RFTS4yMDIwXzEiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiIzOzciLCJhdWQiOlsiNkFDQkQ1MUMtNTE0Qi00RjU5LUIxQzMtQ0IzNUZGN0U5QTZBOzQwMzAwNTkiLCI0Nzc2MWI3NzY1MjJlMTg2ZmNkNzdmMTNjMjVlOGNjZjk5YWY5MDFhNjc4YTY2ZTcwMGIxNjFlZWZlOGZhODhkIl0sInNjb3BlIjpbInJlc3Rfd2Vic2VydmljZXMiLCJyZXN0bGV0cyJdLCJpc3MiOiJodHRwczpcL1wvc3lzdGVtLm5ldHN1aXRlLmNvbSIsIm9pdCI6MTYxMTA2NzY1NSwiZXhwIjoxNjExMDcxMzA0LCJpYXQiOjE2MTEwNjc3MDQsImp0aSI6IjQwMzAwNTkuYS41YjMyMzZiOS1mZmVlLTQyZDMtYmQ1Ny00YmU3YjQ0MzlhMzdfMTYxMTA2NzY1NTM1OS4xNjExMDY3NzA0MTI3In0.TqVYrtJL3hiJwCnAHA4Z067eiVETAvPQhee8s42OZYijEc4eTFXaAhdyo0c0WjnKnhGmH4KvBmcQneTpDAJaiuj2dBDx0LnWxV1tFjhBtxAu2FMlF4rcGi-D7seacZhOo-6szg-oxx25CNv9Bcb1kI9Ly9XI29lazYojIaOaYUVZABurla67boa53BeVMpoqwuitHzFs0VYdeXEWYJ9vpoiaz0aXdZWIPic-wVXoqj3nv4sShrpT4K4-QLQ72gUn1ITWFmOhC_V8SZ5EZH68bEcmwC3cPOZute2-L0AqNMKpiLPt-YD8B85z17dmA9B-hmr77eoGve7zI0UzBNS8iw","expires_in":"3600","token_type":"bearer"}
```

Note: The access token is Base64 encoded. For more information, see RFC 6749. section 1.4.

When the refresh token expires, the token endpoint returns an invalid_grant error. The application must go back to Step One of the OAuth 2.0 authorization code grant flow to restart the process.

Note: Users can use the access token and refresh token to access RESTlets, REST web services, or SuiteAnalytics Connect in case of being locked out.

POST Request to the Revoke Token Endpoint

The application sends the POST request to the revoke token endpoint to revoke the valid refresh token and its associated access tokens.

The format of the URL is:

`https://<accountID>.suitetalk.api.netsuite.com/services/rest/auth/oauth2/v1/revoke`

where <accountID> represents your NetSuite account ID.

Request Parameters for the Revoke Token Request

Request Parameter	Description
token	The value of the token parameter is the value of the refresh token that the application revokes.

Important: The client authentication method used in the header of the request follows the HTTP Basic authentication scheme. For more information, see RFC 7617. The format is **client_id:client_secret**. The string value is Base64 encoded. The following code provides an example.

```
POST /services/rest/auth/oauth2/v1/revoke HTTP/1.1
Host: <accountID>.suitetalk.api.netsuite.com
Authorization: Basic Njc5NGEzMDg2ZTRmNjFhMTIwMzUwZDAxYjg1MjdhZWQzNjMxNDcyZWYzMzQxMjIxMjQ5NWJlNjVhOGZjOGQ0YzpjZGM3YWMyMjE4M2VmNTAyNGU4MWIwZmNlOGVmNDYxYzQ0ZDU4OTZhMWYxODA1ZDRiMzcyY2E2MWM0ZDMyNmFl
Content-Type: application/x-www-form-urlencoded

token=eyJraWQiOiJzLlNZU1RFTS4yMDIwXzEiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiIzOzciLCJhdWQiOlsiNkFDQkQ1MUMtNTE0Qi00RjU5LUIxQzMtQ0IzNUZGN0U5QTZBOzQwMzAwNTkiLCI0Nzc2MWI3NzY1MjJlMTg2ZmNkNzdmMTNjMjVlOGNjZjk5YWY5MDFhNjc4YTY2ZTcwMGIxNjFlZWZlOGZhODhkIl0sIn
Njb3BlIjpbInJlc3Rfd2Vic2VydmljZXMiLCJyZXN0bGV0cyJdLCJpc3MiOiJodHRwczpcL1wvc3lzdGVtLm5ldHN1aXRlLmNvbSIsIm9pdCI6MTYxMTA2NzY1NSwiZX
hwIjoxNjExNjcyNDU1LCJpYXQiOjE2MTEwNjc2NTUsImp0aSI6IjQwMzAwNTkuci41YjMyMzZiOS1mZmVlLTQyZDMtYmQ1Ny00YmU3YjQ0MzlhMzdfMTYxMTA2NzY1NT
M1OS4wIn0.BbSQ86Phg6CKLMJ9gJQurs1NK6niSxFzF2EBFT--KFysI2AV9S1llZgxsRNIrMXsDioaepdWsGzrKepJXq25t5Sr7f-jBwTLK9g9SkFAvvEFsVJCYb
dA4_BNZkHKlCC-1mA_yFNZWBYPdfCMGDX39iIDd7LVkaj-oPjpnuRnnK1ntNzxHx0coJiXwj3KfOI0PK7xfG1zbVSW14XOlatWbi80MY0ZQCgF41nFs-Rv-a7r-b51mMr
m6kKZx-0MXfKRYT60H3gPXCk2QzkKovKy3kBVjajbtVPNS2tF_SbFNWOXJrn4MFzvnnDy0qsxT_Ijy3S5LTgk4YLrlwKv_XoE7A
```

Note: If you use public clients you can choose from the following options:

- The HTTP authorization request header does not contain the Authorization, and the client_id parameter is included in the body of the request, or
- The HTTP authorization request header contains only the client_id in the Authorization.

Integration Record and Prompt Parameter Combinations

See the following table for details about possible combinations of the **Consent Policy** list on the integration record and the prompt parameter in Step One of the OAuth 2.0 code grant flow.

The Consent Policy list value	The prompt parameter value	The Consent Screen
Always Ask	—	The consent screen appears.
Always Ask	none	The consent screen appears.
Always Ask	login	The consent screen appears. A user must authenticate even if there is an active session.
Always Ask	consent	The consent screen appears.
Ask First Time	—	The consent screen appears for the first time. Consent screen also appears if: ■ The consent was not given previously ■ The system does not know which role or account to choose for the user to log in with ■ The application requires a different set of scopes and needs a new consent
Ask First Time	none	The consent screen appears for the first time.
Ask First Time	login	The consent screen appears for the first time. Consent screen also appears if: ■ The consent was not given previously ■ The system does not know which role or account to choose for the user to log in with ■ The application requires a different set of scopes and needs a new consent A user must authenticate even if there is an active session.
Ask First Time	consent	The consent screen appears.
Never Ask	—	The consent screen does not appear.
Never Ask	none	The consent screen does not appear.
Never Ask	login	The consent screen does not appear. A user must authenticate even if there is an active session.
Never Ask	consent	The consent screen appears.

For more information, see Step One GET Request to the Authorization Endpoint and Create Integration Records for Applications to Use OAuth 2.0

OAuth 2.0 Client Credentials Flow

Application developers and integrators can use the client credentials flow with OAuth 2.0. The client credentials flow is machine-to-machine and does not require any user interaction. Administrators and users with the OAuth 2.0 Authorized Applications Management permission can set up the flow and upload certificates for applications on the OAuth 2.0 Client Credentials (M2M) Setup page. Administrators and users with the OAuth 2.0 Authorized Applications Management permission can revoke the certificates on the same page.

The OAuth 2.0 client credentials flow consists of a POST request to the token endpoint and a system response containing an access token.

For more information, see POST Request to the Token Endpoint and the Access Token Response.

POST Request to the Token Endpoint and the Access Token Response

The client credentials flow starts when the application sends a POST request to the token endpoint.

The format of the URL is:

`https://<accountID>.suitetalk.api.netsuite.com/services/rest/auth/oauth2/v1/token`

where <accountID> represents your NetSuite account ID.

POST Request Parameters

Request Parameter	Description
grant_type	The value of the grant_type parameter is always `client_credentials`.
client_assertion_type	The value of the client_assertion_type parameter is always `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`.
client_assertion	The value of the client_assertion parameter is a JWT bearer token. The token is signed with the private part of the certificate used for mapping of the application. For more information about the mapping, see OAuth 2.0 Client Credentials Setup For more information about the JWT bearer token, see The Request Token Structure. **Note:** You should use a library for generating of the JWT bearer token.

Important: Request parameters must be encoded based on the HTML specification for the application/x-www-form-urlencoded media type. For more information, see URL Specification 5.1.

The following example provides a sample POST request:

```
POST /services/rest/auth/oauth2/v1/token HTTP/1.1
Host: <accountID>.suitetalk.api.netsuite.com
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-type%3Ajwt-bearer&client_as
sertion=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiIxYzM0M2E3MTZjMWRjZWI2MGU3ZmMxNDlmYTY3MzU5MjllZjc3ZDI4ZmUxN
jI5M2Y4OTI5NzZkZGU3ZDhlM2UyIiwic2NvcGUiOlsicmVzdGxldHMiLCAicmVzdF93ZWJzZXJ2aWNlcyJdLCJhdWQiOiJodHRwczovL3J1bmJveC5jb3JwLm5ldHN1aXR
lLmNvbS9zZXJ2aWNlcy9yZXN0L2F1dGgvb2F1dGggyL3YxL3Rva2VuIiwiZXhwIjoxNjI3OTA5MzAzLCJpYXQiOjE2Mjc5MDU3MDN9.j7fhtd0qQP-iD7n
s9q_fuG8Arz2aWJyoSvZ8sHRVA8HXOJG3pAQbT5J5F8MLkWIXA9ZuSxHdCWNwQLoRUeKlGURYFFqDHP_yjoWFWWtq5Wb-AnaZg_jBVL8TaOFGY2WByFM8rHsJVopFeg
wEQsU6bkcwqiFttEKxso-MiSAc5lE9SBgi6Fus2btiYGIFcNrKalFXEWDy6Ah5yVCo3wxkk9dfiPmT6JgLdjFkCc3v7tMCD9CrRHXrmhQvL8aoeyTMzJILURw5r
nuy9zAs9ngymtX_iiwes8XpkBeCJbX4totI-EY4myi7L4fc2NgeWT-bvLWo6_sWjXE4BKyewqjtreUJscR9bhJ5Fi7S8nIoGDQbZrwhIgoKM_UI9Waw6kRLwR
er_c0QDFY-sMLeGT3HL5vihHRFNXd-cKb-AWplkRiSJrdHXJtuGHLniHRpkK0-A1AFalIzYw4SSykxfck0qsPdf-oFPuawUsKR9lDCcYlyOaDZdQsBNsbjOsp5gGtyCuB
wPBS8xz7I6gqLVEfNuzTfDDk8SMw1fN9MQ0NJtZMqMxm-WY_bLjZVkI3gqsvgDS-ADBPC7cymVZGfPUqummDUeG-Ks7SkLaHpfY6i-aZS8KUAY4aN5Do3GWT56aoEM9s1Y
B_1ZF_YxsBmK_gcX_mmlwUxbvCVpuHJTvKAQzY
```

The Access Token Response

JSON Response Fields	Description
access_token	The value of the access_token parameter is in JSON Web Token (JWT) format. The access token is valid for 60 minutes.

JSON Response Fields	Description
expires_in	The value of the expires_in parameter is always 3600. The value represents the time period during which the access token is valid, in seconds.
token_type	The value of the token_type parameter is always bearer.

The following is an example of a response in JSON JWT format:


```
{"access_token": "eyJraWQiOiJzLlNZU1RFTS4yMDIwXzEiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiI0MDMwMDU5OzA7Mzs3O04iLCJhdWQiOlsiOWE1MDY4YjFjNGU5OGU4Yjg1YzMwMmYyMjg2N2YzNTAyYTBmYzYwNzU4MDQwNzliNzYzZmExYzg2NzJiYTlkNCIsImFwcDoyNzBDNDQ3Ny1DNUY1LTRFMDQtQkND
MS1CMDMzRDk0QTlDMDgiXSwic2NvcGUiOlsicmVzdF93ZWJzZXJ2aWNlcyIsInJlc3RsZXRzIl0sImlzcyI6Imh0dHBzOlwvXC9ydW5ib3guY29ycC5uZXRzd
Wl0ZS5jb20iLCJvaXQiOjE2Mjc5MDYzMzAsImV4cCI6MTYyNzkwOTkzMCwiaWF0IjoxNjI3OTA2MzMwLCJqdGkiOiJhLmMuZmI3ZDYzN2YtOTdjNC00Nzk0LTkyY
WYtZTU2N2ZhYjc1ODRlLjE2Mjc5MDYzMzA1MDMuMTYyNzkwNjMzMDUwMyJ9.QjzADDeU2yN-6j-ol0fApgmleIn17HHD4bi06yBYpEpL5rBSbK3h11-GgU44Kc6u
jQQQ3t4yr6IWBrtak5qLPWQmJE5-Ry_IvaxZRmPuB8rxI09_o4uXJE7oxpMreK4snYoIfH1Ph40Fq977MVVz9K-5pCTclOberX9dTTM3O0BnL6QNrf3lv3RA7J5Lil
ceGAm4OV7OOoddn_fB6yeO0ZghVbJbRgI-tChqwdmWY42zhTeHjdG4K6ooA2IVcOm2GUFMhiFT2I00ZLZ-dYBPYkfRDn2Fvbn8V8GN1biQ6_u6j07k0XSq1Mv-WN-
saH7rTKaA1gkX4IFwIHzN7eJUcg", "expires_in": "3600", "token_type": "Bearer"}
```


Note: The access token is Base64 encoded. For more information, see RFC 6749. section 1.4.

When the access token expires, the token endpoint returns an invalid_grant error. The application must restart the flow.

The Request Token Structure

The JWT bearer token for the POST request to the token endpoint in OAuth 2.0 client credentials flow includes three parts: a header, a payload, and a signature.

The token header includes the following parameters:

Parameter Name	Description
typ	The value of the typ parameter is always JWT.
alg	The value of the alg parameter is PS256, PS384, PS512, ES256, ES384, or ES512. The value you choose determines the algorithm used for signing of the token.
kid	The value of the kid parameter is the value of the Certificate ID generated during the mapping of the application. For more information, see OAuth 2.0 Client Credentials Setup.

The token payload includes the following parameters:

Parameter Name	Description
iss	The value of the iss parameter is the client ID for the integration. For more information, see Create Integration Records for Applications to Use OAuth 2.0.
scope	The value of the scope parameter is restlets, rest_webservices, suite_analytics, or all of them, separated by a comma.
aud	The value of the aud parameter is the NetSuite token endpoint, https://<accountID>.suitetalk.api.netsuite.com/services/rest/auth/oauth2/v1/token.
exp	The value of the exp parameter represents the number of seconds since January 1, 1970, until the token's expiration. The value must not be more than one hour greater than the value of the iat parameter.

Parameter Name	Description
iat	The value of the iat parameter represents when the token was issued. The value of the parameter is in seconds, since January 1, 1970.

Note: The token's dot-separated values are Base64 encoded.

The public part of the certificate is used to validate the signature. The private part of the same certificate is used in the kid parameter.

OAuth 2.0 Access and Refresh Token Structure

Both the access token and refresh token include three parts: a header, a payload, and a signature.

The token header includes the following parameters:

Parameter Name	Description
kid	The value of the kid parameter is the ID of the certificate used for signing the token.
typ	The value of the type parameter is `JWT`.
alg	The value of the alg parameter is `RS256`.

The token payload includes the following parameters:

Parameter Name	Description
sub	The value of the sub parameter is the role and entity of the user, separated by a semicolon. For example, 1111;10.
aud	The value of the aud parameter is the integration record and the company, separated by a semicolon. Additionally, the client ID is a part of the aud parameter, separated by a comma. For example, 1A111AA1–AA11–1A11–1111–A1A1111111A1;1111, 661131f7bf0a2f8a4d09c79d3fa961eab66102dca43e07ad47d3a29628ced67b.
scope	The value of the scope parameter is either `restlets`, `rest_webservices`, `suite_analytics`, or all of them, separated by a comma.
iss	The value of the iss parameter is `https://system.netsuite.com`.
oit	The value of the oit parameter represents the number of seconds since the first token of the token chain was issued. This is only applicable for public clients.
exp	The value of the exp parameter represents the number of seconds since January 1, 1970, until the token's expiration.
iat	The value of the iat parameter represents when the token was issued. The value of the parameter is in seconds, since January 1, 1970.
jti	The value of the jti parameter is the token ID, which is unique for every token.

Note: The token's dot-separated values are Base64 encoded.

The signature is validated with a public key, which is associated with the kid parameter. To access the public key for your account, use the following URL:

`https://<accountID>.suitetalk.api.netsuite.com/services/rest/auth/oauth2/v1/keys`

where <accountID> represents your NetSuite account ID.

Certificate Rotation

As of NetSuite 2021.1, the certificates used to validate access and refresh tokens during the OAuth 2.0 code grant flow are no longer valid indefinitely.

The certificates are valid for 90 days and the system generates new certificates 30 days before the previous certificates expire.

The certificates are company-specific.

Troubleshooting OAuth 2.0

Note: Applications authorized using the OAuth 2.0 feature in your NetSuite production account are not copied to your Release Preview account or to your sandbox accounts. Users must authorize applications explicitly in Release Preview and sandbox accounts to test the OAuth 2.0 feature in these accounts. Each time a sandbox account is refreshed, users again must authorize applications explicitly in the sandbox account.

See the following topics for OAuth 2.0 troubleshooting information:

- Authorization Code Grant Flow Errors
  - Authorization Errors in Step One
  - Response Errors in Step Two and in the Refresh Token Response
  - RESTlets and REST Web Services Authentication Errors
- OAuth 2.0 and the Login Audit Trail
  - Tracking OAuth 2.0 Integrations and Users
  - RESTlets and REST Web Services Error Messages in the Login Audit Trail
  - Authorization Code Grant Flow Error Messages in the Login Audit Trail
  - The Refresh Token Request Error Messages in the Login Audit Trail
- OAuth 2.0 Authorization Header Examples
  - RESTlet Authorization Header
  - REST Web Services Authorization Header

Authorization Code Grant Flow Errors

For information about errors that may occur during the OAuth 2.0 flow, see the following topics:

- Authorization Errors in Step One
- Response Errors in Step Two and in the Refresh Token Response
- RESTlets and REST Web Services Authentication Errors

Authorization Errors in Step One

The following table lists errors that may occur in Step One of the OAuth 2.0 authorization code grant flow. Error requests are sent to the redirect URI with a specific error value, and should be handled by the application.

The redirect parameter is error.

Error Value	Error Description	Resolution
invalid_request	One or more required parameters are missing. **Important:** The redirect does not take place if the redirect URI in the GET request does not match the value in the **Redirect URI** field in the corresponding integration record. Only the error message should be displayed.	Ensure that none of the parameters is missing in the request in Step One. For more information, see Step One GET Request to the Authorization Endpoint.
unauthorized_client	The redirect does not take place if the client is unknown to the authorization server. Only the error message should be displayed.	Ensure that you use the correct values of the client ID and client secret for the corresponding integration record.
access_denied	A user clicks the **Deny** or **Back** button on the consent screen and interrupts the flow.	The user must Click **Allow** or **Continue** to give the application consent. For more information, see Consent Screen.
unsupported_response_type	The response type cannot be handled.	Ensure that the response type value is correct. For more information, see Step One GET Request to the Authorization Endpoint.
invalid_scope	The scope cannot be handled. The scope value is malformed, unknown, or invalid.	Ensure that the scope value is in correct format. For more information, see Step One GET Request to the Authorization Endpoint.

The following is an example of a redirect to the redirect URI with an error:

```
https://<your_redirect_uri>?
state=ykv2XLx1BpT5Q0F3MRPHb94j&role=1000&entity=12&company=1234567&error=<error_value>
```

For more information about Step One of the OAuth 2.0 authorization code grant flow, see Step One GET Request to the Authorization Endpoint.

Response Errors in Step Two and in the Refresh Token Response

The following table lists errors that may occur in Step Two of the OAuth 2.0 authorization code grant flow and in the response to the refresh token request.

The JSON format for the response is:

```
{
    "error": "<error_value>"
}
```

Error Value	Error Description	Resolution
invalid_request	Any of the following conditions can cause the invalid_request error to occur: ■ One or more required parameters are missing or malformed. ■ The grant_type value is incorrect. ■ Multiple client authentication approaches are used. ■ Any other type of a malformed request is sent. The HTTP status code is 400 Bad Request.	Ensure that your request is valid and in the correct format. For more information, see Step Two POST Request to the Token Endpoint.

invalid_client	Authentication of the client fails.
	The HTTP status code is 401 Unauthorized.
	The response header is set to:
	`Basic realm=<accountID>`
	Following is an example of the response header:
	`HTTP/1.1 401 Unauthorized`
	`WWW-Authenticate: Basic realm="123456"`
		Ensure that you use the correct values of the client ID and client secret for the corresponding integration record.

invalid_grant	Any of the following conditions can cause the invalid_grant error to occur: ■ The authorization code is invalid, expired, or revoked. ■ The refresh token is invalid, expired, or revoked. **Important:** In case the refresh token is expired, the application must go back to Step One of the OAuth 2.0 authorization code grant flow to restart the process. ■ The redirect URI does not match the redirect URI in the authorization request. ■ The authorization code or refresh token cannot be associated with the client. ■ The code_verifier parameter on Step Two does not match the code_verifier parameter in Step One. The HTTP status code is 400 Bad Request.	Ensure that values of all parameters are correct and matching the values from Step One of the flow. For more information, see Step Two POST Request to the Token Endpoint.
unauthorized_client	The value of the authorization grant_type is not allowed for the client.	Ensure that you use the correct values of the client ID and client secret for the corresponding integration record.

Error Value	Error Description	Resolution
unsupported_grant_type	The value of the grant_type parameter is neither `authorization_code` nor `refresh_token`. The HTTP status code is 400 Bad Request.	Ensure that the value of grant_type parameter is `authorization_code` or `refresh_token`.
invalid_scope	The scope cannot be handled. The scope value is malformed, unknown, or invalid. The HTTP status code is 400 Bad Request.	Ensure that the scope value is in correct format. For more information, see Step Two POST Request to the Token Endpoint.

For more information about Step Two of the OAuth 2.0 authorization code grant flow, see Step Two POST Request to the Token Endpoint.

For more information about the refresh token request, see Refresh Token POST Request to the Token Endpoint.

- OAuth 2.0
- OAuth 2.0 Tasks for Administrators
- OAuth 2.0 for Integration Application Developers
- OAuth 2.0 Authorization Code Grant Flow
- Troubleshooting OAuth 2.0
- Authorization Code Grant Flow Errors

RESTlets and REST Web Services Authentication Errors

The following table lists WWW-Authenticate response header errors that may occur when RESTlets and REST web services are authenticated.

Error Value	Error_description Value	Error Description
invalid_request	The request could not be understood by the server due to malformed syntax.	The request is in a wrong format. One or more parameters are missing, repeated, or malformed. The HTTP status code is 400 Bad Request.
invalid_token	Invalid login attempt.	The provided access token is expired, revoked, malformed, or invalid. The HTTP status code is 401 Unauthorized.

The following examples show headers for the errors in the preceding table:

```
HTTP/1.1 400 Bad Request
      WWW-Authenticate: Bearer realm="123456",
            error="invalid_request",
            error_description="The request could not be understood by the server due to malformed syntax."
```

```
HTTP/1.1 401 Unauthorized
      WWW-Authenticate: Bearer realm="123456",
            error="invalid_token",
            error_description="Invalid login attempt."
```

Note: The value of the realm is the account ID for which the data are requested.

- OAuth 2.0
- OAuth 2.0 Tasks for Administrators

- OAuth 2.0 for Integration Application Developers
- OAuth 2.0 Authorization Code Grant Flow
- Troubleshooting OAuth 2.0
- Authorization Code Grant Flow Errors

OAuth 2.0 and the Login Audit Trail

This section covers how to use the Login Audit Trail to track integrations and users, and provides details about error messages you may encounter.

For more information about tracking integrations and users, see Tracking OAuth 2.0 Integrations and Users.

For more information about error messages, see the following sections:

- RESTlets and REST Web Services Error Messages in the Login Audit Trail
- Authorization Code Grant Flow Error Messages in the Login Audit Trail
- The Refresh Token Request Error Messages in the Login Audit Trail

Tracking OAuth 2.0 Integrations and Users

You can use the Login Audit Trail to track OAuth 2.0 integrations and users.

To track integrations 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** and **Token-based Application Name**.
5. Click **Submit**.

   The **Detail** column displays error messages for any OAuth 2.0 logins with a status of Failure.

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

RESTlets and REST Web Services Error Messages in the Login Audit Trail

The following table lists errors that are visible in the **Detail** column of the Login Audit Trail Results.

Problem	RESTlets/REST Web Services	Resolution
The access token is expired.	AccessTokenExpired	Use the refresh token to get a new access token. If the refresh token is expired, initiate the authorization code grant flow to get a new pair of tokens. For more information, see OAuth 2.0 Authorization Code Grant Flow.
At least one of the following is invalid: - Entity - Contact - Role	EntityOrRoleDisabled	Verify that the entity, contact, or role exists in the account.

Problem	RESTlets/REST Web Services	Resolution
The signature is invalid.	InvalidSignature	Ensure that you use the correct public key for token validation. For more information, see OAuth 2.0 Access and Refresh Token Structure. **Warning:** Invalidity of issuer or signature may be caused by cross-site request forgery (CSFR) attacks. To ensure that your application is safe, follow the OAuth 2.0 specification. For more information, see RFC6749 Section 10.12.
Login attempted with a refresh token.	TokenRejected	Ensure that the application uses the access token for access and the refresh token for the refresh token POST request. For more information, see Refresh Token POST Request to the Token Endpoint.
The integration application ID is invalid.	InvalidIntegration	Verify that the corresponding integration record exists in the account.
The integration application has empty scope or the scope in the token does not match the scope in the integration record.	ScopeMismatched	Ensure that the **RESTlets** or **REST Web Services** box is checked in the corresponding integration record. For more information, see Create Integration Records for Applications to Use OAuth 2.0.
The integration application does not use OAuth 2.0.	AuthorizationCodeGrantRequired	Ensure that the **Authorization Code Grant** box is checked in the corresponding integration record. For more information, see Create Integration Records for Applications to Use OAuth 2.0.
The scope value is empty in the token.	InvalidScope	Ensure that the structure of the access token is correct. For more information, see OAuth 2.0 Access and Refresh Token Structure.
Role or entity is inactive.	EntityOrRoleDisabled	Verify that the entity or role is active in the account.
The OAuth 2.0 feature is not enabled in the account.	FeatureDisabled	See Enable the OAuth 2.0 Feature.
The integration record is blocked.	IntegrationBlocked	Ensure that the value of the **State** field is set to **Enabled** on the corresponding integration record. For more information, see Create Integration Records for Applications to Use OAuth 2.0.

Authorization Code Grant Flow Error Messages in the Login Audit Trail

The following table lists errors that are visible in the **Detail** column of the Login Audit Trail Results.

Problem	Authorization Code Grant Flow Step One	Authorization Code Grant Flow Step Two	Resolution
The integration application has empty	ScopeMismatched	ScopeMismatched	Ensure that either the **RESTlets** or **REST Web Services** box is

Problem	Authorization Code Grant Flow Step One	Authorization Code Grant Flow Step Two	Resolution
scope or the scope in the token does not match the scope in the integration record.			checked in the corresponding integration record. For more information, see Create Integration Records for Applications to Use OAuth 2.0.
The integration application does not use OAuth 2.0.	AuthorizationCodeGrant Required	AuthorizationCodeGrant Required	Ensure that the **Authorization Code Grant** box is checked in the corresponding integration record. For more information, see Create Integration Records for Applications to Use OAuth 2.0.
Role or entity is inactive.	—	EntityOrRoleDisabled	Verify that the entity or role is active in the account.
The value of the state parameter is invalid.	InvalidState	—	Ensure that the value of the state parameter: ■ is 24 to 1024 characters long ■ consists of printable ASCII characters
Client ID or client secret is invalid.	UnknownIntegration	ClientAuthenticationFailed	Ensure that you use the correct values of the client ID and client secret for the corresponding integration record.
The value of the redirect URI parameter is invalid.	InvalidRedirectURI	—	Ensure that the redirect URI is a valid URL. for more information, see Create Integration Records for Applications to Use OAuth 2.0.
The response type is invalid.	UnsupportedResponseType	—	The response type used is not valid for this step of the authorization code grant flow. For more information, see Step One GET Request to the Authorization Endpoint.
The user clicked Deny/ Back on the consent screen.	AuthorizationExplicitly Denied	—	Start the OAuth 2.0 authorization code grant flow again and click Allow/Continue on the consent screen.
The value of the grant type parameter is either invalid or wrong.	—	InvalidGrantType	Ensure that the grant type value used is the correct one in the corresponding step of the authorization code grant flow. For more information, see OAuth 2.0 Authorization Code Grant Flow.
The OAuth 2.0 feature is not enabled in the account.	FeatureDisabled	FeatureDisabled	See Enable the OAuth 2.0 Feature.

Problem	Authorization Code Grant Flow Step One	Authorization Code Grant Flow Step Two	Resolution
The integration record is blocked.	IntegrationBlocked	IntegrationBlocked	Ensure that the value of the **State** field is set to **Enabled** on the corresponding integration record. For more information, see Create Integration Records for Applications to Use OAuth 2.0.
Parameters for the Proof Key for Code Exchange (PKCE) are missing or malformed	InvalidRequest	—	If you use PKCE in OAuth 2.0, make sure you configured the parameters correctly. For more information, see Step One GET Request to the Authorization Endpoint.
The code_verifier parameter in Step Two does not match the code_verifier parameter in Step One.	—	InvalidGrant	If you use PKCE in OAuth 2.0, make sure you configured the parameters correctly. For more information, see Step One GET Request to the Authorization Endpoint, and Step Two POST Request to the Token Endpoint.

The Refresh Token Request Error Messages in the Login Audit Trail

The following table lists errors that are visible in **Detail** column of the Login Audit Trail Results.

Problem	Refresh Token Request	Resolution
The access token is expired.	RefreshTokenExpired	Use the refresh token to get a new access token. If the refresh token is expired, initiate the authorization code grant flow to get a new pair of tokens. For more information, see OAuth 2.0 Authorization Code Grant Flow.
At least one of the following is invalid: ■ Entity ■ Contact ■ Role	EntityOrRoleDisabled	Verify that the entity, contact, or role exists in the account.
The signature is invalid.	InvalidSignature	Ensure that you use the correct public key for token validation. For more information, see OAuth 2.0 Access and Refresh Token Structure.

Problem	Refresh Token Request	Resolution
		Warning: Invalidity of issuer or signature may be caused by cross-site request forgery (CSFR) attacks. To ensure that your application is safe, follow the OAuth 2.0 specification. For more information, see RFC6749 Section 10.12.
The integration application ID is invalid.	InvalidIntegration	Verify that the corresponding integration record exists in the account.
The integration application has empty scope or the scope in the token does not match the scope in the integration record.	ScopeMismatched	Ensure that either the **RESTlets** or **REST Web Services** box is checked in the corresponding integration record. For more information, see Create Integration Records for Applications to Use OAuth 2.0.
The integration application does not use OAuth 2.0.	AuthorizationCodeGrantRequired	Ensure that the **Authorization Code Grant** box is checked in the corresponding integration record. For more information, see Create Integration Records for Applications to Use OAuth 2.0.
The scope value is empty in the token.	InvalidScope	Ensure that the structure of the access token is correct. For more information, see OAuth 2.0 Access and Refresh Token Structure.
Role or entity is inactive.	EntityOrRoleDisabled	Verify that the entity or role is active in the account.
Client ID or client secret is invalid.	ClientAuthenticationFailed	Ensure that you use the correct values of the client ID and client secret for the corresponding integration record.
The value of the grant type parameter is invalid or wrong.	InvalidGrantType	Ensure that the grant type value used is the correct one in the corresponding step of the authorization code grant flow. For more information, see OAuth 2.0 Authorization Code Grant Flow.
The application attempted the refresh token request with an access token.	InvalidRefreshToken	Ensure that the application uses the access token for accessing RESTlets and REST web services, and the refresh token for the refresh token POST request. For more information, see Refresh Token POST Request to the Token Endpoint.
The OAuth 2.0 feature is not enabled in the account.	FeatureDisabled	See Enable the OAuth 2.0 Feature.
The integration record is blocked.	IntegrationBlocked	Ensure that the value of the **State** field is set to **Enabled** on the corresponding integration record. For more information, see Create Integration Records for Applications to Use OAuth 2.0.

OAuth 2.0 Authorization Header Examples

For information about authorization headers for RESTlets and REST web services, see the following topics:

- RESTlet Authorization Header
- REST Web Services Authorization Header

RESTlet Authorization Header

The URL format for the RESTlet authorization header is:

`https://<accountID>.app.netsuite.com/app/site/hosting/restlet.nl?script=1&deploy=1`

The structure of the authorization header is:

`Authorization: Bearer <access_token>`

The following is an example of the OAuth 2.0 authorization header for RESTlets:

```
Authorization: Bearer eyJraWQiOiIyMDIwXzEiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiIxMDAwOzEyIiwiYXVkIjoiN0VCODkwREMtNEJDR
C00RTQ5LTkzNDEtRjZEMDIyNDUxOEY5OzM4Mjk4NTUiLCJ0dHlwZSI6IkFDQ0VTUyIsInNjb3BlIjpbIlJFU1RMRVRTIl0sImlzcyI6Imh0dHBzOlwvXC9zeXN0ZW0ub
mV0c3VpdGUuY29tIiwiZXhwIjoxNTgwODI1NjQyLCJpYXQiOjE1ODA4MjIwNDJ9.sTNSUlE1w-X_zhNPou_pRvHPob_p6iTkvA329yfVqrFFcgy0Ma14HA1WtlYmd8Xy8T
GvC5str_ZYEBNq9adNSb1inkgB4orFCus5plvCzuLaeA_kYWc6KEFq6Z2jfBBymrDtLqujvvBMxNan88KN0UXM7CaNDGrg7tUllcQcB6mJwiqrRMXPWPXSZMc17C
groIPwvNCaF7mK9np4V-s0nhlCCII_XuESWXZom2nJtserwiLC7db2psrmtXKSu0l75XRYWb8Qn1G3x56oYz56TAfjB2bM6kUYq-s4Io2QHHdD0HxZSH-d_i5gY3s
fCIqzr9Z4G8u6IHLN0fThDTt3hQ
```

For more information, see Setting up OAuth 2.0 for a RESTlet Integration.

REST Web Services Authorization Header

The URL format for the REST web services authorization header is:

`https://<accountID>.suitetalk.api.netsuite.com/services/rest/record/v1/customer`

The structure of the authorization header is:

`Authorization: Bearer <access_token>`

The following is an example of the OAuth 2.0 authorization header for REST web services:

```
Authorization: Bearer eyJraWQiOiIyMDIwXzEiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiIxMDAwOzEyIiwiYXVkIjoiN0VCODkwREMtNEJDR
C00RTQ5LTkzNDEtRjZEMDIyNDUxOEY5OzM4Mjk4NTUiLCJ0dHlwZSI6IkFDQ0VTUyIsInNjb3BlIjpbIlJFU1RMRVRTIl0sImlzcyI6Imh0dHBzOlwvXC9zeXN0ZW0ub
mV0c3VpdGUuY29tIiwiZXhwIjoxNTgwODI1NjQyLCJpYXQiOjE1ODA4MjIwNDJ9.sTNSUlE1w-X_zhNPou_pRvHPob_p6iTkvA329yfVqrFFcgy0Ma14HA1WtlYmd8Xy8T
GvC5str_ZYEBNq9adNSb1inkgB4orFCus5plvCzuLaeA_kYWc6KEFq6Z2jfBBymrDtLqujvvBMxNan88KN0UXM7CaNDGrg7tUllcQcB6mJwiqrRMXPWPXSZMc17C
groIPwvNCaF7mK9np4V-s0nhlCCII_XuESWXZom2nJtserwiLC7db2psrmtXKSu0l75XRYWb8Qn1G3x56oYz56TAfjB2bM6kUYq-s4Io2QHHdD0HxZSH-d_i5gY3s
fCIqzr9Z4G8u6IHLN0fThDTt3hQ
```

For more information, see the help topic Setting Up OAuth 2.0 Authentication for REST Web Services.

OAuth 2.0 for RESTlets

The following details about using OAuth 2.0 with RESTlets are provided here for your convenience.

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
- Setting up OAuth 2.0 for a RESTlet Integration
- Using OAuth 2.0 for RESTlet Authentication

Important: For information about OAuth 2.0 authorization header for RESTlets, see OAuth 2.0 Authorization Header.

For information about related tasks, see the following topics:

- Create Integration Records for Applications to Use OAuth 2.0
- Regenerating a Consumer Key and Secret

Note: Applications authorized using the OAuth 2.0 feature in your NetSuite production account are not copied to your Release Preview or to your sandbox accounts. Users must authorize applications explicitly in Release Preview or a sandbox account to test OAuth 2.0 feature in these accounts. Each time the sandbox is refreshed, users must authorize applications explicitly in the sandbox account.

OAuth 2.0 for REST Web Services

OAuth 2.0 is only available for REST web services and RESTlets. SOAP web services do not support OAuth 2.0.

For more information, see the following topics:

- Setting Up OAuth 2.0 Authentication for REST Web Services
- OAuth 2.0 Authorization Code Grant Flow
- OAuth 2.0 Authorization Header for REST Web Services

For information about related tasks, see the following topics:

- Create Integration Records for Applications to Use OAuth 2.0
- Regenerating a Consumer Key and Secret

Note: Applications authorized using the OAuth 2.0 feature in your NetSuite production account are not copied to your Release Preview or to your sandbox accounts. Users must authorize applications explicitly in Release Preview or a sandbox account to test OAuth 2.0 feature in these accounts. Each time the sandbox is refreshed, users must authorize applications explicitly in the sandbox account.

OAuth 2.0 Authorization Header for REST Web Services

After you finish the authorization code grant flow and the application is granted an access token, see the following information to create the OAuth 2.0 authorization header.

The format of URL is:

```
https://<accountID>.suitetalk.api.netsuite.com/services/rest/record/v1/customer
```

The structure of the authorization header is:

```
Authorization: Bearer <access token>
```

The following is an example of the OAuth 2.0 authorization header for REST web services:

```
Authorization: Bearer eyJraWQiOiIyMDIwXzEiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiIxMDAwOzEyIiwiYXVkIjoiN0VCODkwREMtNEJDRC00RTQ5LTkzNDEtRjZEMDIyNDUxOEY5OzM4Mjk4NTUiLCJ0dHlwZSI6IkFDQ0VTUyIsInNjb3BlIjpbIlJFU1RMRVRTIl0sImlzcyI6Imh0dHBzOlwvXC9zeXN0ZW0ubmV0c3VpdGUuY29tIiwiZXhwIjoxNTgwODI1NjQyLCJpYXQiOjE1ODA4MjIwNDJ9.sTNSUlE1w-X_zhNPou_pRvHPob_p6iTkvA329yfVqrFFcgy0Ma14HA1WtlYmd8Xy8TGvC5str_ZYEBNq9adNSb1inkgB4orFCus5plvCzuLaeA_kYWc6KEFq6Z2jfBBymrDtLqujvvBMxNan88KN0UXM7CaNDGrg7tUllcQcB6mJwiqrRMXPWPXSZMc17CgroIPwvNCaF7mK9np4V-s0nhlCCII_XuESWXZom2nJtserwiLC7db2psrmtXKSu0l75XRYWb8Qn1G3x56oYz56TAfjB2bM6kUYq-s4Io2QHHdD0HxZSH-d_i5gY3sfCIqzr9Z4G8u6IHLN0fThDTt3hQ
```

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. 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 want to receive verification codes when they set up their 2FA preferences. When users set up their 2FA preferences, they also receive a list of one-time use backup codes.
  - An authentication application that complies with OATH TOTP. The app generates a time-based verification code for each login. Authenticator apps are supported in all NetSuite accounts. Users should select an authenticator app as the primary method of authentication. See the help topic Supported Authenticator Apps.
  - A phone that can receive verification codes by Short Message Service (SMS) message or by voice call. SMS and voice call are subject to carrier availability and changes in local regulations. Delivery of verification codes by SMS or voice call is not as reliable as using an authenticator app.

> **Important:** As of **March 1, 2023**, users setting up or resetting their 2FA configurations must install and use an authenticator app to generate verification codes. Receiving codes by SMS will no longer be supported for users setting up 2FA for the first time, or for existing users who reset their 2FA settings. Users can also log in with the one-time backup codes provided during 2FA setup.
>
> The SMS option is currently only prohibited for new 2FA setups. However, industry experts (such as NIST, W3C, and the FIDO Alliance) no longer view SMS as a secure delivery mechanism for 2FA verification codes. As part of our ongoing commitment to world-class security, the SMS option is targeted for removal in a future NetSuite release.

See the following sections for more information:

- What Administrators Need to Know About 2FA
- Benefits of 2FA in Your NetSuite Account

To read 2FA help topics available to users, see the help topic Logging In Using Two-Factor Authentication (2FA).

What Administrators Need to Know About 2FA

- Certain roles with highly privileged permissions require 2FA in NetSuite. 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 or OAuth 2.0. See Token-based Authentication (TBA) and OAuth 2.0 for more information.

> **Note:** OAuth 2.0 is only available for use with RESTlets and REST web services. It cannot be used with SOAP web services.

- 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:

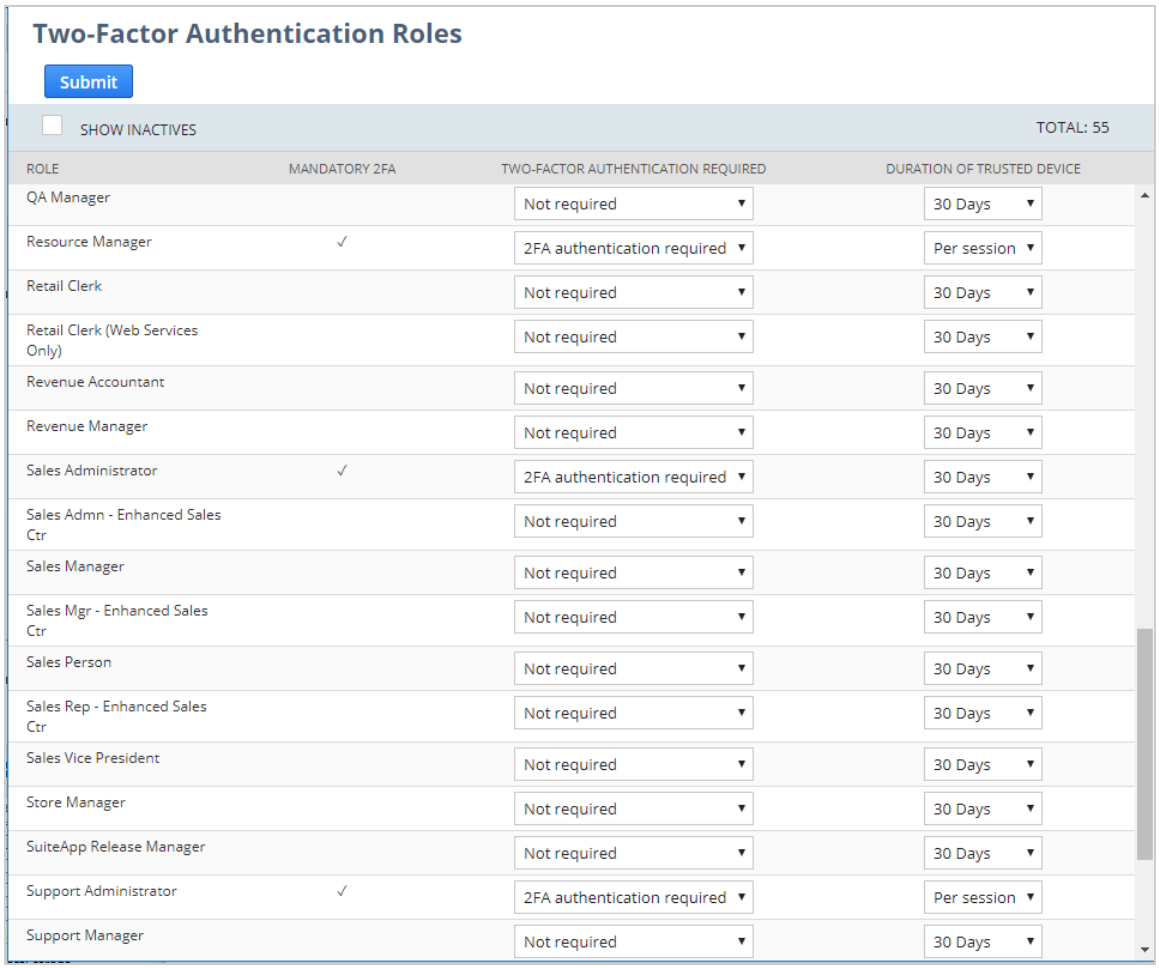
- No special licensing is required. (No cost.)
- No special tokens are required. (No cost.)
- Access is supported for the NetSuite UI and NetSuite Mobile applications.
- Little maintenance is required of administrators. After being assigned to a 2FA authentication required role, users configure their 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 provide 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. SMS message and voice call are not reliable due to dependence on mobile signal, international restrictions, or roaming. 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 about 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. 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 required for the Administrator role and other roles with highly privileged permissions. These roles are indicated in Mandatory 2FA column 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).



Required 2FA Tasks

See the following required tasks for managing two–factor authentication (2FA) in a NetSuite account. These tasks can be completed by 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 their 2FA preferences. Authenticator apps for generating 2FA verification codes are supported in all NetSuite accounts. Users should select an authenticator app as the primary method of authentication. In the setup wizard, user can then skip to backup codes, or can select SMS or Voice Call as the secondary method. See the following for help written for users: Set up Your Preferences for Two-Factor Authentication (2FA).

- To generate verification codes using an Authenticator App, users must install an authenticator application.
- To receive verification codes by phone, users must register a phone number in NetSuite, which is tied to the user's email address. SMS and voice call are subject to carrier availability and changes in local regulations. Delivery of verification codes by SMS or voice call is not as reliable as using an authenticator app.

  Important: As of **March 1, 2023**, users setting up or resetting their 2FA configurations must install and use an authenticator app to generate verification codes. Receiving codes by SMS will no longer be supported for users setting up 2FA for the first time, or for existing users who reset their 2FA settings.

  The SMS option is currently only prohibited for new 2FA setups. However, industry experts (such as NIST, W3C, and the FIDO Alliance) no longer view SMS as a secure delivery mechanism for 2FA verification codes. As part of our ongoing commitment to world-class security, the SMS option is targeted for removal in a future NetSuite release.
- Users are provided ten backup codes, to be used when they are not able to obtain a verification code through their preferred methods.

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 also enter a verification code. Each verification code is a unique series of numbers valid for a limited time, and only for a single login. During 2FA 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.

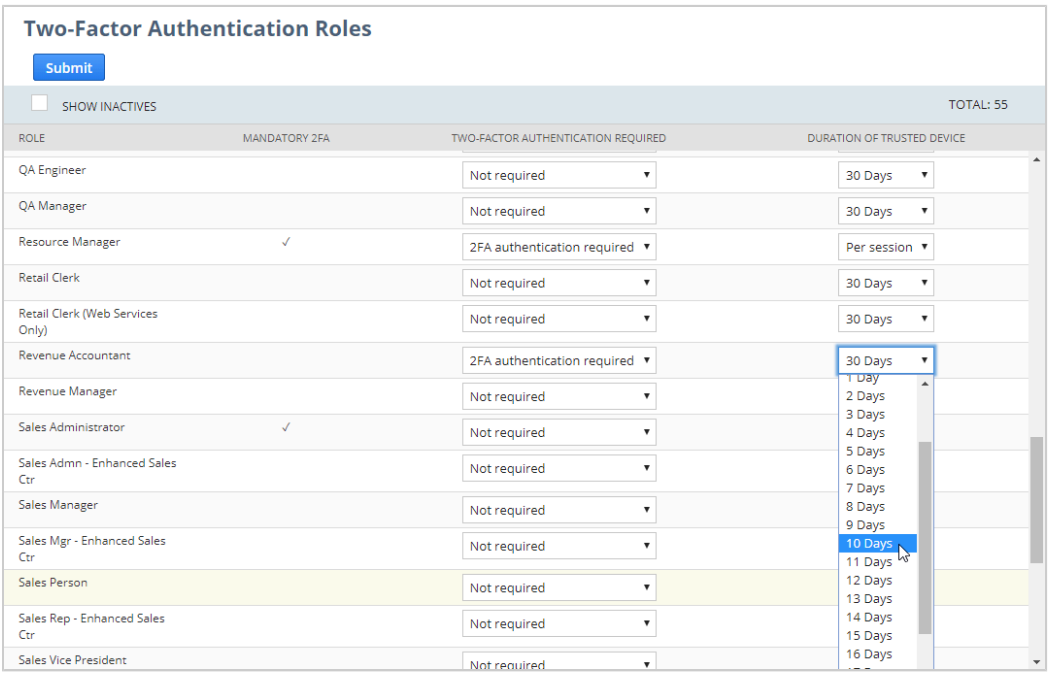
An administrator or another user with the Two-Factor Authentication base permission can use the Two-Factor Authentication Roles page to indicate roles that require 2FA for login. Each 2FA role can be configured to specify how often users with that role should be presented with the 2FA challenge. The 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.

2FA is required 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).

To designate two-factor authentication roles:

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 or OAuth 2.0. See Token-based Authentication (TBA) or OAuth 2.0 for more information. OAuth 2.0 cannot be used with SOAP web services.

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

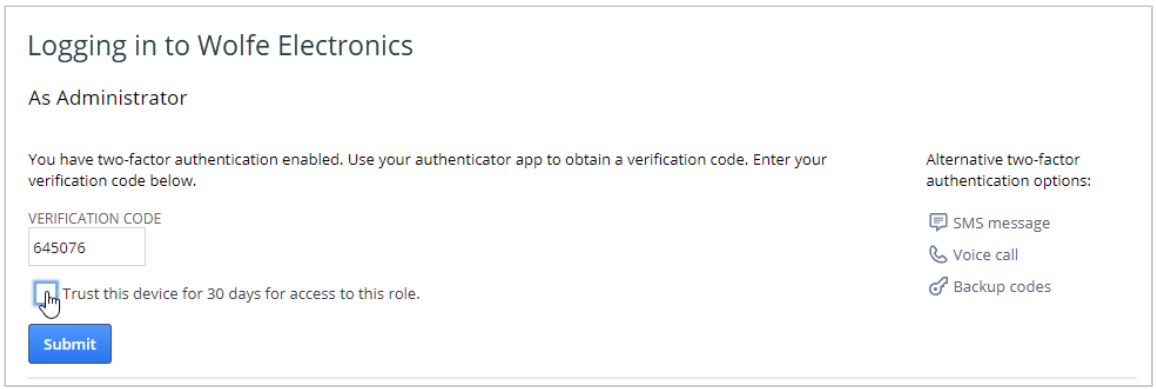
Users and Trusted Devices for Two-Factor Authentication

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.

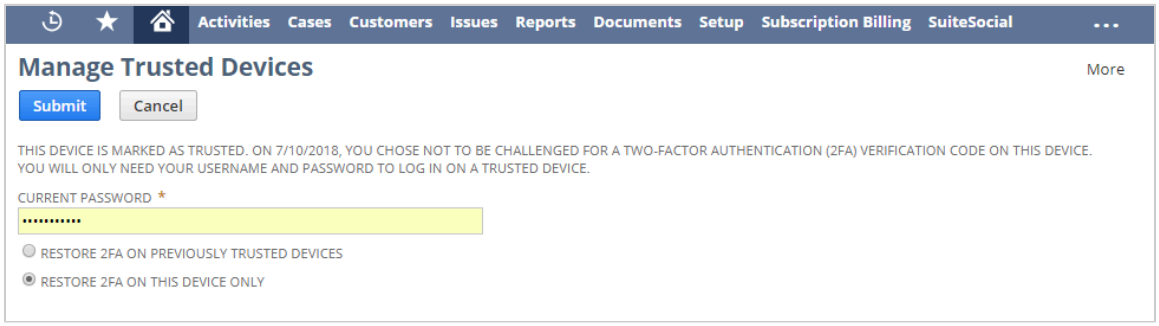


The user is in complete control of whether devices are considered trusted. In this example, the user could check the Trust this device box and then would not be presented with a 2FA challenge for 30 days when logging in to NetSuite from this device.

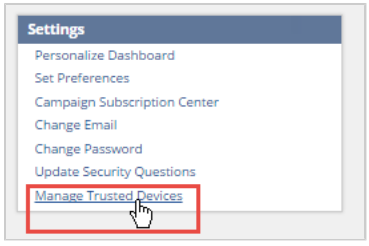
After a user has marked a device as trusted, the user can modify that choice on the Manage Trusted Devices page.

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

The following two videos about using 2FA in NetSuite are available.

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. If 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 various companies.

Authenticator apps for generating 2FA verification codes are supported in all NetSuite accounts. Users should select an authenticator app as the primary method of authentication.

Important: As of **March 1, 2023**, users setting up or resetting their 2FA configurations must install and use an authenticator app to generate verification codes. Receiving codes by SMS will no longer be supported for users setting up 2FA for the first time, or for existing users who reset their 2FA settings. Users can also log in with the one-time backup codes provided during 2FA setup.

For more information, see:

The SMS option is currently only prohibited for new 2FA setups. However, industry experts (such as NIST, W3C, and the FIDO Alliance) no longer view SMS as a secure delivery mechanism for 2FA verification codes. As part of our ongoing commitment to world-class security, the SMS option is targeted for removal in a future NetSuite release.

To use 2FA, 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

The User Access Reset Tool is available to administrators so that they can reset 2FA settings for users. It may be necessary for an administrator to reset 2FA settings for users who may be unable to log in and reset them on their own.

You can reset both the password, and the 2FA settings, or only one of those.

Important: To initiate a password reset for a user who has access to multiple NetSuite accounts, you must be an administrator in all of those accounts. The User Access Reset Tool is also available to users with Core Administration Permissions, but the same restriction applies. Users who are not administrators but have only Core Administration Permissions cannot reset the password for a user who has access to multiple NetSuite accounts. For more information, see the help topic Core Administration Permissions.

User Access Reset Tool

See the following procedure for information about using the User Access Reset tool to reset 2FA settings.

To reset a user's 2FA settings with the User Access Reset Tool:

1. In an Administrator role, or a role with Core Administration Permissions (CAP), go to Setup > Users/Roles > User Management > User Access Reset Tool.

   For more information about CAP, see the help topic Core Administration Permissions.
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.
   a. **Initiate Password Reset**: Check this box to send an email to the user containing a link so that the user can reset the NetSuite password.
   b. **Clear User's 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.
   c. **Unlock The User's Access**: Check this box to unlock NetSuite access for a user who is locked out of NetSuite after submitting five consecutive incorrect passwords.
   d. **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**.

For more information, see User Access Reset Tool.

Supported Countries: SMS and Voice Call

Important: As of **March 1, 2023**, users setting up or resetting their 2FA configurations must install and use an authenticator app to generate verification codes. Receiving codes by SMS will no longer be supported for users setting up 2FA for the first time, or for existing users who reset their 2FA settings. Users can also log in with the one-time backup codes provided during 2FA setup.

The SMS option is currently only prohibited for new 2FA setups. However, industry experts (such as NIST, W3C, and the FIDO Alliance) no longer view SMS as a secure delivery mechanism for 2FA verification codes. As part of our ongoing commitment to world-class security, the SMS option is targeted for removal in a future NetSuite release.

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.

Country	Supported Delivery Methods	Country Code
Afghanistan	SMS	+93
Åland Islands (Finland)	SMS Voice call	+358
Albania	SMS Voice call	+355
Algeria	SMS	+213
American Samoa	SMS	+1 Area Code: 684
Andorra	SMS Voice call	+376
Angola	SMS	+244
Anguilla	SMS	+1 Area Code: 264
Antigua and Barbuda	SMS	+1 Area Code: 268
Argentina	SMS Voice call	+54
Armenia	SMS Voice call	+374
Aruba	SMS	+297
Ascension	SMS	+247
Australia	SMS Voice call	+61
Austria	SMS Voice call	+43
Azerbaijan	SMS Voice call	+994
Bahamas	SMS	+1 Area Code: 242
Bahrain	SMS	+973
Bangladesh	SMS	+880
Barbados	SMS	+1 Area Code: 246

Country	Supported Delivery Methods	Country Code
Belarus	SMS Voice call	+375
Belgium	SMS Voice call	+32
Belize	SMS	+501
Benin	SMS	+229
Bermuda	SMS	+1 Area Code: 441
Bhutan	SMS	+975
Bolivia	SMS	+591
Bosnia and Herzegovina	SMS Voice call	+387
Botswana	SMS	+267
Brazil	SMS Voice call	+55
British Virgin Islands	SMS Voice call	+1 Area Code: 284
Brunei	SMS	+673
Bulgaria	SMS Voice call	+359
Burkina Faso	SMS	+226
Burundi	SMS	+257
Cambodia	SMS	+855
Cameroon	SMS	+237
Canada	SMS Voice call	+1 Multiple Area Codes
Canary Islands	SMS	+3491
Cape Verde	SMS	+238
Caribbean Netherlands (Netherlands Antilles)	SMS	+599
Cayman Islands	SMS	+1 Area Code: 345
Central African Republic	SMS	+236

Country	Supported Delivery Methods	Country Code
Chad	SMS	+235
Chile	SMS Voice call	+56
China	Authenticator app only. SMS is not available. Voice call is not available.	+86
Christmas Island	SMS Voice call	+61
Cocos (Keeling) Islands	SMS Voice call	+61
Colombia	SMS	+57
Comoros	SMS	+269
Congo, Democratic People's Republic	SMS	+243
Congo, Republic of	SMS	+242
Costa Rica	SMS	+506
Côte d'Ivoire (Ivory Coast)	SMS	+225
Croatia	SMS Voice call	+385
Cuba	SMS	+53
Cyprus	SMS Voice call	+357
Czech Republic	SMS Voice call	+420
Denmark	SMS Voice call	+45
Djibouti	SMS	+253
Dominica	SMS	+1 Area Code: 767
Dominican Republic	SMS	+1 Area Code: 809
East Timor	SMS	+670
Ecuador	SMS	+593
Egypt	SMS Voice call	+20

Country	Supported Delivery Methods	Country Code
El Salvador	SMS	+503
Equatorial Guinea	SMS	+240
Eritrea	SMS	+291
Estonia	SMS Voice call	+372
Ethiopia	SMS	+251
Falkland Islands	SMS	+500
Faroe Islands	SMS Voice call	+298
Fiji	SMS Voice call	+679
Finland (including the Åland Islands)	SMS Voice call	+358
France	SMS Voice call	+33
French Guiana	SMS	+594
French Polynesia	SMS	+689
Gabon	SMS	+241
Gambia	SMS	+220
Georgia	SMS Voice call	+995
Germany	SMS Voice call	+49
Ghana	SMS	+233
Gibraltar	SMS	+350
Greece	SMS Voice call	+30
Greenland	SMS	+299
Grenada	SMS	+1 Area Code: 437
Guadeloupe	SMS	+590
Guam	SMS	+1 Area Code: 671

Country	Supported Delivery Methods	Country Code
Guatemala	SMS	+502
Guernsey (United Kingdom)	SMS Voice call	+44
Guinea	SMS	+224
Guinea-Bissau	SMS	+245
Guyana	SMS	+592
Haiti	SMS Voice call	+509
Honduras	SMS	+504
Hong Kong	SMS Voice call	+852
Hungary	SMS Voice call	+36
Iceland	SMS Voice call	+354
India	SMS Voice call	+91
Indonesia	SMS Voice call	+62
Iran	SMS	+98
Iraq	SMS	+964
Ireland	SMS Voice call	+353
Isle of Man (United Kingdom)	SMS Voice call	+44
Israel	SMS Voice call	+972
Italy	SMS Voice call	+39
Jamaica	SMS	+1 Area Code: 876
Japan	SMS Voice call	+81
Jersey	SMS	+44

Country	Supported Delivery Methods	Country Code
(United Kingdom)	Voice call	
Jordan	SMS	+962
Kazakhstan	SMS Voice call	+7
Kenya	SMS	+254
Kosovo	SMS	+883
Kuwait	SMS Voice call	+965
Kyrgyzstan	SMS Voice call	+996
Laos	SMS	+856
Latvia	SMS Voice call	+371
Lebanon	SMS	+961
Lesotho	SMS	+266
Liberia	SMS	+231
Libya	SMS	+218
Liechtenstein	SMS Voice call	+423
Lithuania	SMS Voice call	+370
Luxembourg	SMS Voice call	+352
Macau	SMS Voice call	+853
Macedonia	SMS Voice call	+389
Madagascar	SMS	+261
Malawi	SMS	+265
Malaysia	SMS Voice call	+60
Maldives	SMS	+960
Mali	SMS	+223
Malta	SMS	+356

Country	Supported Delivery Methods	Country Code
	Voice call	
Marshall Islands	SMS	+692
Martinique	SMS	+596
Mauritania	SMS	+222
Mauritius	SMS	+230
Mayotte	SMS	+262
Mexico	SMS Voice call	+52
Micronesia	SMS	+691
Moldova	SMS Voice call	+373
Monaco	SMS Voice call	+377
Mongolia	SMS	+976
Montenegro	SMS Voice call	+382
Montserrat	SMS	+1 Area Code: 664
Morocco	SMS	+212
Mozambique	SMS	+258
Myanmar	SMS	+95
Namibia	SMS	+264
Nepal	SMS	+977
Netherlands	SMS Voice call	+31
Netherlands Antilles (Caribbean Netherlands)	SMS	+599
New Caledonia	SMS	+687
New Zealand	SMS Voice call	+64
Nicaragua	SMS	+505
Niger	SMS	+227
Nigeria	SMS	+234
North Korea	SMS	+850

Country	Supported Delivery Methods	Country Code
Northern Mariana Islands	SMS	+1 Area Code: 670
Norway	SMS Voice call	+47
Oman	SMS	+968
Pakistan	SMS Voice call	+92
Palau	SMS	+680
Palestinian Territory (Palestine)	SMS	+970
Panama	SMS Voice call	+507
Papua New Guinea	SMS	+675
Paraguay	SMS Voice call	+595
Peru	SMS Voice call	+51
Philippines	SMS Voice call	+63
Poland	SMS Voice call	+48
Portugal	SMS Voice call	+351
Puerto Rico	SMS Voice call	+1 Area Codes: 787, 939
Qatar	SMS Voice call	+974
Réunion Island	SMS	+262
Romania	SMS Voice call	+40
Russia (Russian Federation)	SMS Voice call	+7
Rwanda	SMS	+250
Saint Barthélemy	SMS	+590
Saint Kitts and Nevis	SMS	+1

Country	Supported Delivery Methods	Country Code
		Area Code: 869
Saint Lucia	SMS	+1 Area Code: 758
Saint Martin (French side)	SMS	+590
Saint Pierre and Miquelon	SMS	+508
Saint Vincent and the Grenadines	SMS	+1 Area Code: 784
Samoa	SMS	+685
San Marino	SMS Voice call	+378
São Tomé and Principé	SMS	+239
Saudi Arabia	SMS Voice call	+966
Senegal	SMS	+221
Serbia	SMS Voice call	+381
Seychelles	SMS	+248
Sierra Leone	SMS	+232
Singapore	SMS Voice call	+65
Slovakia (Slovak Republic)	SMS Voice call	+421
Slovenia	SMS Voice call	+386
Solomon Islands	SMS	+677
Somalia	SMS	+252
South Africa	SMS Voice call	+27
South Korea	SMS Voice call	+82
South Sudan	SMS	+211
Spain	SMS Voice call	+34

Country	Supported Delivery Methods	Country Code
Sri Lanka	SMS	+94
Sudan	SMS	+249
Suriname	SMS	+597
Svalbard and Jan Mayen (Norway)	SMS Voice call	+47
Swaziland	SMS	+268
Sweden	SMS Voice call	+46
Switzerland	SMS Voice call	+41
Syria (Syrian Arab Republic)	SMS	+963
Taiwan	SMS Voice call	+886
Tajikistan	SMS Voice call	+992
Tanzania	SMS	+255
Thailand	SMS Voice call	+66
Timor-Leste (East Timor)	SMS	+670
Togo	SMS	+228
Tonga	SMS	+676
Trinidad and Tobago	SMS	+1 Area Code: 868
Tunisia	SMS	+216
Turkey	SMS Voice call	+90
Turkish Republic of Northern Cyprus	SMS	+90
Turkmenistan	SMS Voice call	+993
Turks and Caicos Islands	SMS	+1 Area Code: 649
Tuvalu	SMS	+688

Country	Supported Delivery Methods	Country Code
U.S. Virgin Islands	SMS Voice call	+1 Area Code: 340
Uganda	SMS	+256
Ukraine	SMS Voice call	+380
United Arab Emirates	SMS Voice call	+971
United Kingdom	SMS Voice call	+44
United States	SMS Voice call	+1 Multiple Area Codes
Uruguay	SMS Voice call	+589
Uzbekistan	SMS Voice call	+998
Vanuatu	SMS	+678
Vatican City	SMS Voice call	+379
Venezuela	SMS Voice call	+58
Vietnam	SMS Voice call	+84
Virgin Islands, British	SMS Voice call	+1 Area Code: 284
Virgin Islands, U.S.	SMS Voice call	+1 Area Code: 340
Western Sahara	SMS	+212
Yemen	SMS	+967
Zambia	SMS	+260
Zimbabwe	SMS	+263

Device ID Authentication

Device ID Authentication allows administrators to restrict login to only approved devices. Devices can be registered in NetSuite using a unique identifier. After reviewing registered devices, the 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 administrators is required to use the device record.

See the following for information about using this feature:

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 about SCIS, see the help topic SuiteCommerce InStore Administrator's Overview. See also, Configuring the SCIS Mobile App.

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

Administrators can create additional device ID restricted roles if preferred, using the SCIS-created roles as a template. For more information about 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 an administrator reviews the device record and changes the device status to **Trusted**. This requirement ensures that only devices approved by the administrator can be used to log in. The 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, administrators maintain and manage the list of the POS devices that can potentially access the SCIS website in your NetSuite account.

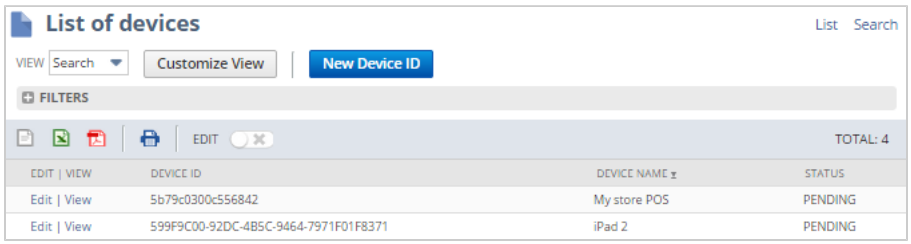
The List of devices page

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 appears in full.

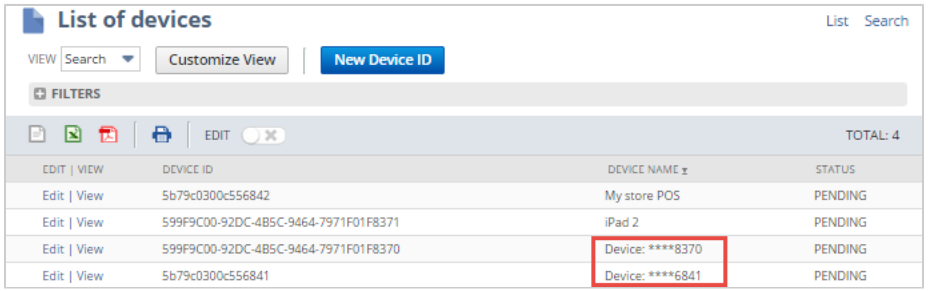
Important: To allow the administrator the opportunity to verify the device, the device ID appears in full. As soon as the administrator changes the status, the device ID is masked and cannot be retrieved from the system.



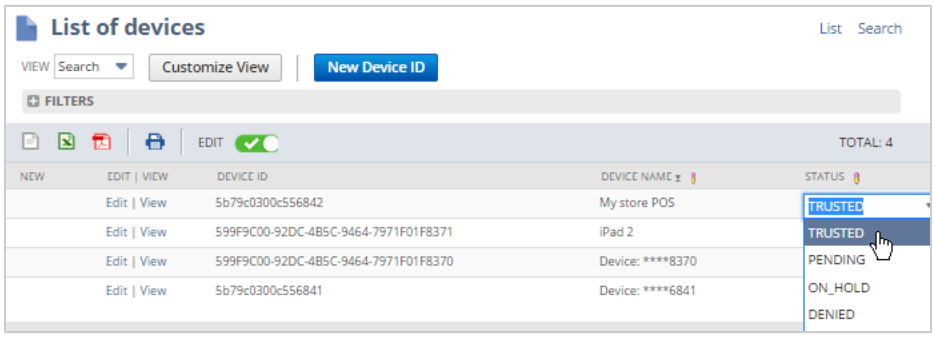
Note: After the device record has been created, you should not change the device name on the device. If the name is changed on the device, administrators would need to make the corresponding update to the device record in NetSuite. The device record in NetSuite is never updated by the POS application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

In the following screenshot, the 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.



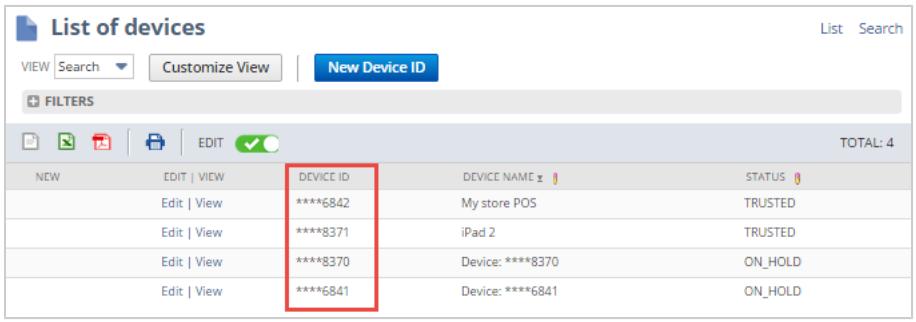
The following screenshot is an example of the administrator reviewing the List of devices, and changing the status of each device. The administrator is sure the Device IDs for the automatically created records are correct, and changes the status to Trusted.



For the manually created records, the 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 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

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

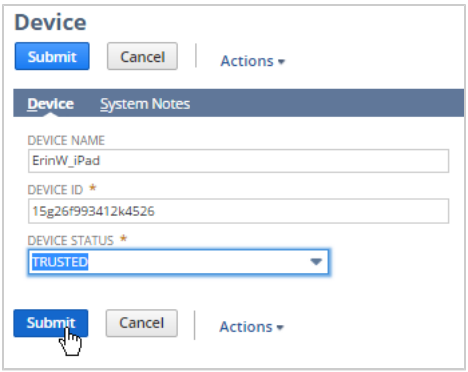
It is possible for administrators to create device records manually, but there is the potential for data entry errors. Before creating the record, ensure you have access to the correct device ID and device name.

To create a device record:

1. Go to Setup > Integration > Device ID.
2. Click **New Device ID**. The **Device** tab appears 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 preferred. Devices in any status other than Trusted will not be allowed to log in to NetSuite.

   - **Pending** - Awaiting review and approval from an administrator. For device records created manually, the administrator should change 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 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 administrator and denied access to NetSuite.
6. Click **Submit**.

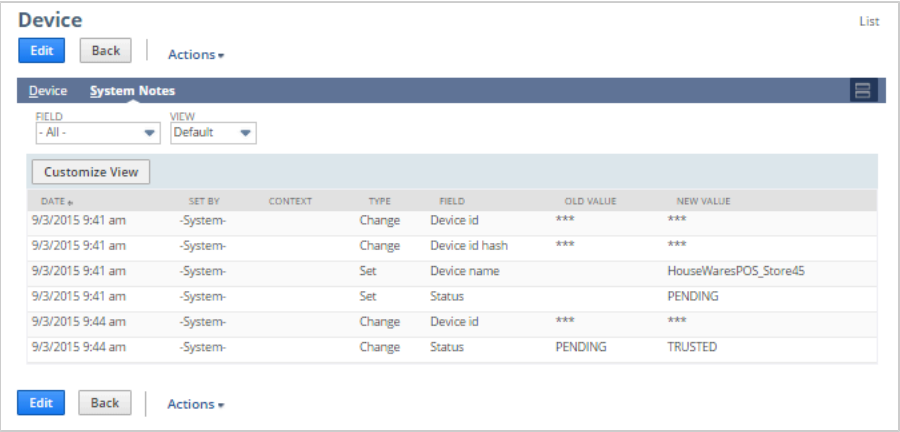


Viewing System Notes

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.



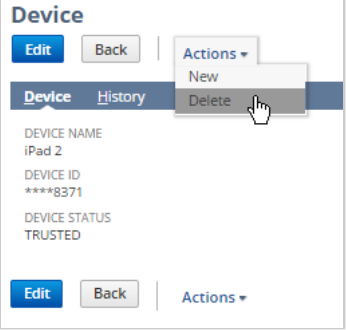
For more information about system notes, see the help topic System Notes Overview.

Deleting a Device Record

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**.

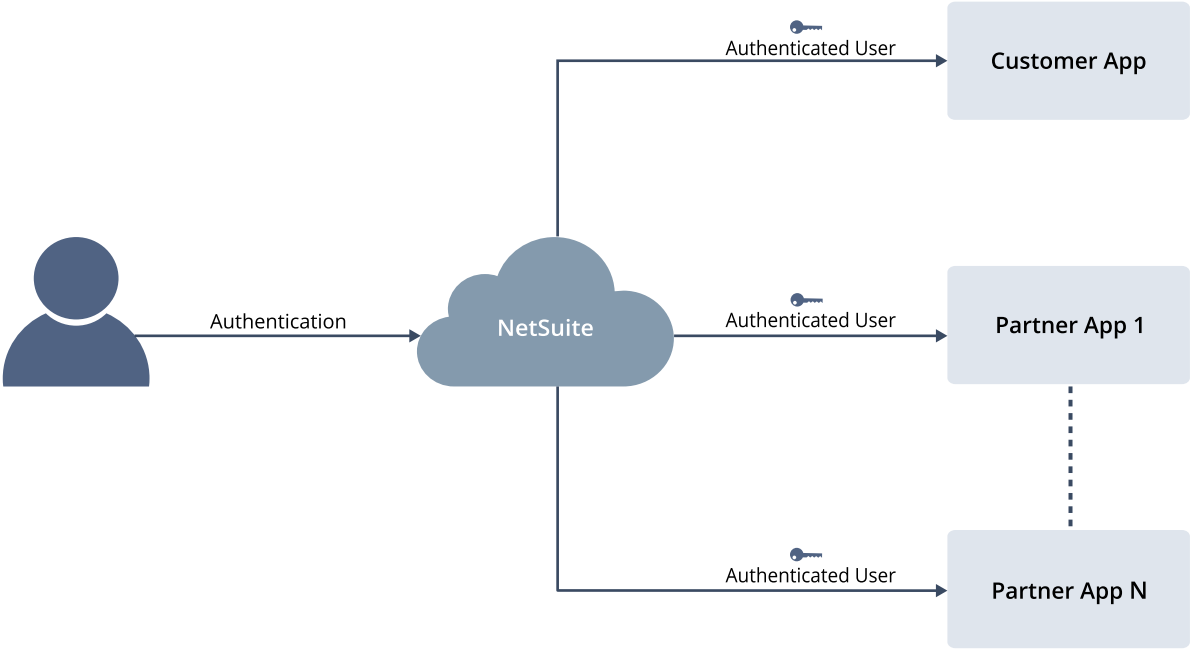
   A confirmation notice appears on the List of devices page.

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:

- OpenID Connect (OIDC) Single Sign-on
- SAML 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.x.
- 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 various 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 appears 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

Warning: As of March 1, 2021, the dc and env parameters in the Outbound SSO HTTP call is no longer supported. You must start using the systemDomain and webservicesDomain parameters instead. For more information, see step 3 in SuiteSignOn Connection Details section.

See the following sections for information about SuiteSignOn.

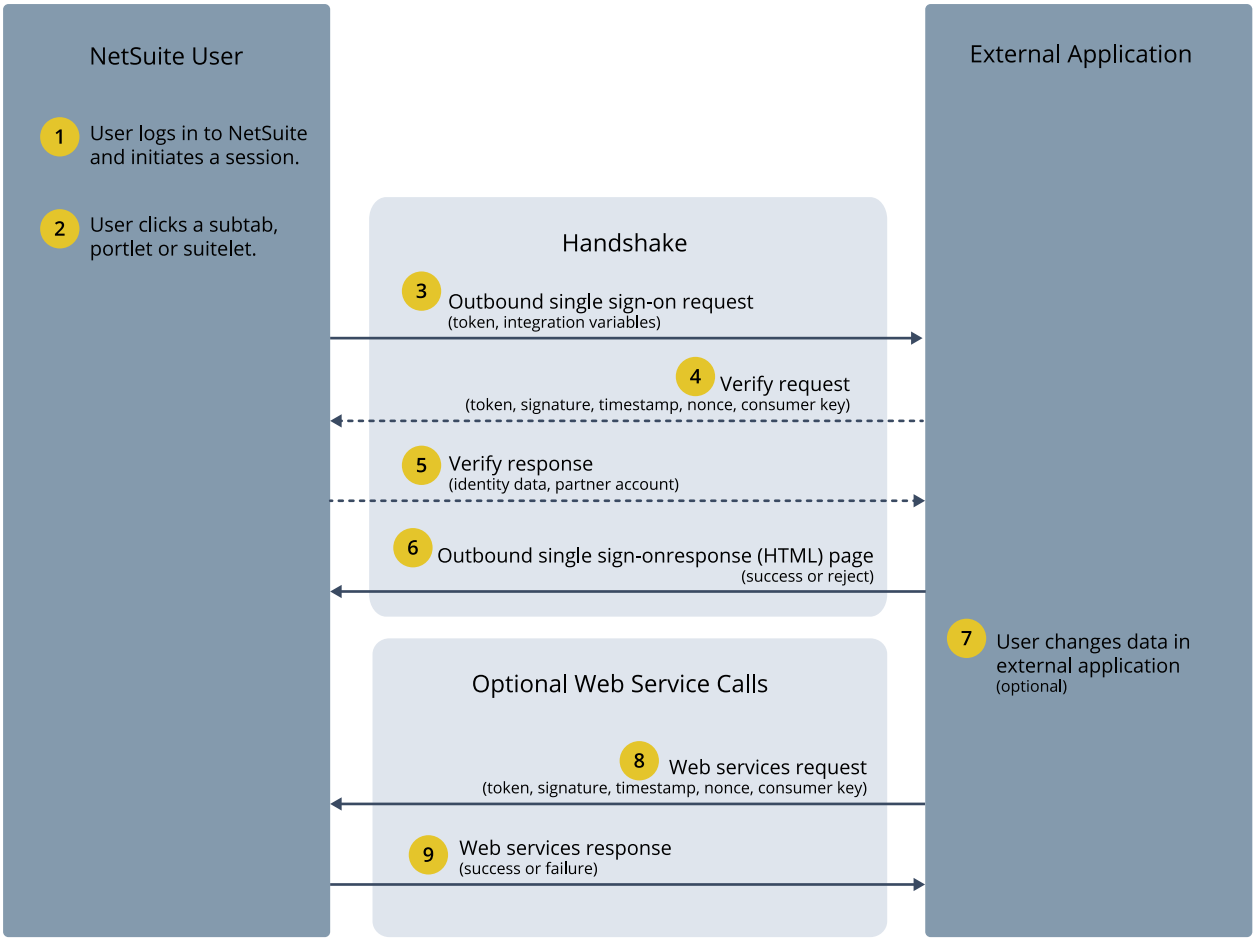
- SuiteSignOn Sequence Diagram

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 systemDomain URL parameter, and a webservicesDomain URL parameter for the optional web services calls. Send the verify request to the domain specified by the value of the systemDomain parameter. Send the web services request to the domain specified by the value of the webservicesDomain parameter. If any data fields were previously defined as required context for the connection, NetSuite sends values for these fields at the same time.

   Warning: The outbound HTTP call still includes a dc and an env URL parameters, but you should not use them. Using the hard-coded mapping between the dc parameter and the URL might cause problems when your account is moved to a different data center that is missing in your mapping.
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 about 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, see 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 appears. Or, if there is a problem, an error is returned instead.

   Important: 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 following features can be enabled at Setup > Company > Enable Features, on the SuiteCloud subtab.

Feature	Required for Application Providers?	Required for SuiteSignOn User Accounts?	Notes
SuiteSignOn	Yes	Yes	—
SuiteTalk (web services)	Maybe	Maybe	Required if SuiteSignOn code includes SOAP web services calls.
SuiteBundler	Yes	No	Required to create and deploy bundles. Not required to install bundles.
Client and Server SuiteScript	Maybe	Maybe	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.

Setting Up SuiteSignOn Integration

Important: You should use NetSuite as OIDC Provider as an outbound single sign-on method for use with integrations. For more information, see NetSuite as OIDC Provider.

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:

1. Enable the SuiteSignOn feature and other required SuiteCloud features in your NetSuite account. See SuiteSignOn Required Features.
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 administrators to enter or import values for these fields as needed.

   See Using Custom Fields as SuiteSignOn User Identification.
5. Create a NetSuite SuiteSignOn record. See Creating SuiteSignOn Records.
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 **Edit** 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 finish 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 appear 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:

- **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 appears 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 appear in the connection point. This page must be secure, with an https:// URL. SuiteSignOn is not supported for http:// sites. You can specify a unique 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=`
  - 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 show 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 varying Record Type for each.

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

Defining Integration Variables for Connection Points

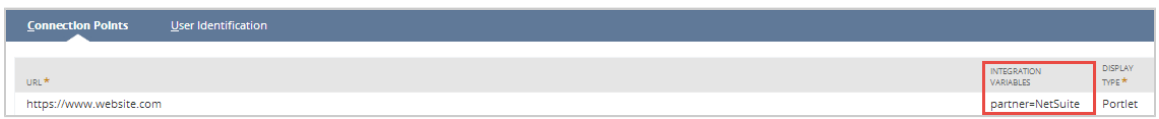
You can define both static and dynamic integration variables for portlet, Suitelet, and user event connection points.

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.

```
/**
 *@NApiVersion 2.x
 *@NScriptType Portlet
 */
define(['N/runtime','N/sso'], function(runtime, sso) {
        function render(context) {
            context.portlet.title = 'My Integrated Application!!';
            var email = runtime.getCurrentUser().email;
            var url = sso.generateSuiteSignOnToken({suiteSignOnId: 'customsso_myApp'});
            url = url + '&partner=NetSuite' + '&email=' + email;
            var content = '<iframe src="'+url+'" align="center" style="width: 100%;height: 600px; margin:0;border:0;padding:0"></
iframe>';
            context.portlet.html = content;
        }
        return {
            render: render
        };
});
```

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.

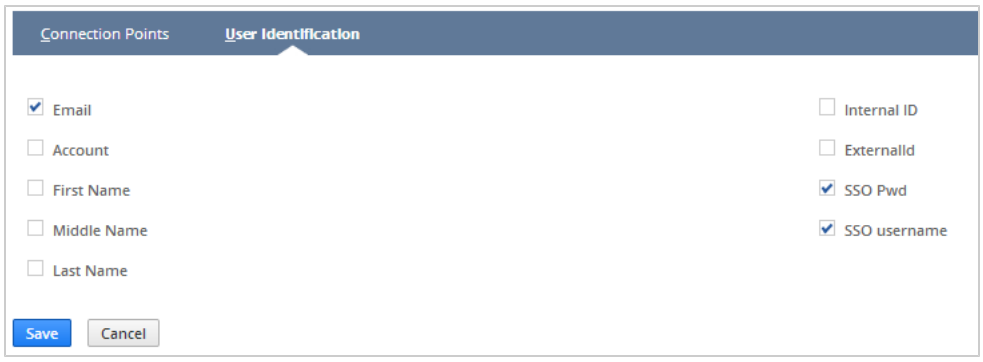
The following user identification fields are provided on the SuiteSignOn page:

- **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.



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.

Only the following field types are supported:

- Check Box
- Currency
- Decimal
- Email Address
- Free-Form Text
- Hyperlink
- Integer
- Percent
- Phone Number
- Text Area

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

You must determine a way for 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 create 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 the 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 various 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

  Important: Only a Suitelet connection point is supported for SuiteSignOn access from your web store.
- Creating a User Event Connection Point

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 appear 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:

Functionality	Subtab	Portlet	Suitelet	User Event
Required NetSuite customizations	Creation of a custom subtab. No scripting required.	Custom scripting required.	Custom scripting required.	Custom scripting required.
Availability on dashboard	Visible within specifically defined records. Automatically available after bundle installed.	Can be added to any page that allows custom portlets. Not available until custom portlet is added and configured to display script.	Link can be added to any page menu. Automatically available after bundle installed.	Not exposed in user interface. Connection is initiated either Before Load, Before Submit, or After Submit, based on the user event script record function.

Functionality	Subtab	Portlet	Suitelet	User Event
Ability to modify application "look and feel"	Limited. External application landing page appears 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 appear on transaction records such as sales order, cash sale, opportunity
- **Entity** - Can appear on entity records such as customer, vendor, employee
- **Item** - Can appear on item records such as inventory, non-inventory, assembly/bill of materials
- **CRM** - Can appear on CRM records such as task, phone call, event
- **Custom Record Subtab** - Can appear 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 **Edit**, 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 needed, 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 appears 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 appear 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 various 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.

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

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 source account, and inform bundle users of the change, so they can update their installations to get the latest version.

To edit a SuiteSignOn record:

1. Go to Setup > Integration > SuiteSignOn, and click **Edit** for a record.
2. Make changes as needed. 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

Note: The SuiteSignOn records in a sandbox account that were copied from your production account after the sandbox refresh, does not have the **Change ID** button.

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 do not appear, 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:

Object	When to include in SuiteSignOn Bundle
SuiteSignOn Outbound Connection	Always 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.

Object	When to include in SuiteSignOn Bundle
Custom Field(s)	If integration uses custom fields as integration variables Custom fields defined as user identification are automatically added.
Saved CSV Import	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

You also should include bundle documentation, a file that provides instructions for 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 the account ID of the bundle's source account.

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 administrators' choices.

Making SuiteSignOn Integrations Available to Users

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: 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 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 the source account for the bundle. To install a bundle from an account, you must have 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 **Documentation**.

You should follow the instructions in this file. The following list 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: 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 for Roles.

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. 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 various 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 **Customize this Page**.
2. In the **Add Content** panel, drag a **Custom Portlet** object to the required location on the page.
3. In the **Custom Content** portlet, click **Set Up**.
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 https://oauth.net/1/.

See the following sections for more information:

- NetSuite SuiteSignOn Translation of OAuth Definitions
- 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:

NetSuite Term	Definition	Analogous OAuth Term
Service Provider	NetSuite	Service Provider
Consumer	External application provider of the application to be accessed from NetSuite through SuiteSignOn, may also be known as **partner**.	Consumer
Consumer Key	Globally unique identifier of the consumer, generated by NetSuite.	Consumer Key
Shared Secret	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.	Consumer Secret

NetSuite Term	Definition	Analogous OAuth Term
Token	Value used to gain access to protected resources on behalf of the user, generated by NetSuite, good for a single session.	—
User	Individual who has logged into NetSuite and initiated activity between the consumer and NetSuite.	User

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=05016d16126a7a6c554656421e2423100608070 51b17ee54e6d26986d8aa&dc=001&env=PRODUCTION&systemDo
main=https%3A%2F%2F<accountID>.app.netsuite.com&webserviceDomain=https%3A%2F%2F<webservicesdomain>app.netsuite.com HTTP/1.1
Host: externalsystem.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:19.0) Gecko/20100101 Firefox/19.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

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 systemDomain and an webservicesDomain parameter. The values of the systemDomain and webservicesDomain parameters are provided by NetSuite.

  Warning: The outbound HTTP call still includes a dc and an env URL parameters, but you should not use them. Using the hard-coded mapping between the dc parameter and the URL might cause problems when your account is moved to a different data center that is missing in your mapping.
- 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=05016d16126a7a6c554656421e2423100608070 51b17ee54e6d26986d8aa
&customer_id=970&dc=001&env=PRODUCTION&systemDomain=https%3A%2F%2F<accountID>.app.netsuite.com&webserviceDomain=https%3A%2F
%2F<webservicesdomain>app.netsuite.com 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.

```
GET /app/common/integration/ssoapplistener.nl HTTP/1.0
Host: <accountID>.app.netsuite.com
Authorization: OAuth oauth_consumer_key="6OtBtQV4nmEOQKpw", oauth_to
ken="05016d16126a7a6c554656421e242310060807051b17ee54e6d26986d8aa", oauth_nonce="kPeHzQpN6bZXsWu5w2nm", oauth_timestam
p="1490706743", oauth_signature_method="HMAC-SHA256", oauth_version="1.0", oauth_signature="vh3C69af9EwXKGbmlDqeA4xiYb
taM1Mq9WH60it4e5Q%3D"
```

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 account specific domain.
- 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.

Field	Description
oauth_token	The token generated and sent by NetSuite.
oauth_consumer_key	A globally unique identifier for the application provider, generated by NetSuite when the integration is set up on the SuiteSignOn page.
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. Refer to the OAuth specification. Go to https://tools.ietf.org/html/rfc5849#section-3.4. See Generate the Signature for the OAuth Header for Outbound SSO for more information.

Field	Description
	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. The shared secret is 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. For more information about the shared secret, see Notes about Modifying the Shared Secret.
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:

```
$url = "https://<accountID>.app.netsuite.com/app/common/integration/ssoapplistener.nl"
$oauth_consumer_key="6OtBtQV4nmEOQKpw"
$oauth_consumer_secret= "P@ssw0rd 123"; //shared secret
$oauth_token="030f6c1d1b6b106c6b445655477e72571343502efefc809d"
$oauth_nonce="kPeHzQpN6bZXsWu5w2nm"
$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 encoded 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="030f6c1d1b6b106c6b445655477e72571343502efefc809d", oauth_consumer_key="6OtBtQV4nmEO
QKpw", oauth_nonce="kPeHzQpN6bZXsWu5w2nm", oauth_timestamp="1490706743", oauth_signature_method="HMAC-SHA256", oauth_ver
sion="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=<accountID>.app.netsuite.com; path=/
X-Powered-By: Servlet/2.5 JSP/2.1
P3P: CP="CAO PSAa OUR BUS PUR"
Vary: User-Agent
Connection: close
Content-Type: text/html; charset=utf-8

<?xml version="1.0" encoding="UTF-8"?>
<outboundSso>
    <entityInfo>
        <ENTITYLASTNAME>Smith</ENTITYLASTNAME>
        <ENTITYINTERNALID>79</ENTITYINTERNALID>
        <ENTITYACCOUNT>1326288</ENTITYACCOUNT>
        <ENTITYFIRSTNAME>John</ENTITYFIRSTNAME>
        <ENTITYEMAIL>jsmith@netsuite.com</ENTITYEMAIL>
    </entityInfo>
</outboundSso>
```

Important: Be aware of the following:

- The domain set in the cookie is the same as the Host value in the external application HTTP verify call.
- The XML element formatting of fields is as name/value pairs, with element names formatted as follows: <ENTITYFIELDID> for standard fields, and <FIELDID> for custom fields.

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

```
WWW-Authenticate: OAuth realm="https%3A%2F%2Facct-java10026.bos.netledger.com", oauth_problem="token_expired"
```

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

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

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 One: Construct a Base String for the Signature
- Step Two: Signature Key
- Step Three: 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). The examples in this section use PHP rawurlencode.

Input Parameters for the Example

These are the input parameters used for this example.

```
$url = 'https://<accountID>.app.netsuite.com/app/common/integration/ssoapplistener.nl';
$httpMethod = 'GET';
$tokenKey = '030e6a121766126c6b445655477e7252517c395926f3430a';
$tokenSecret = '';                    //Outbound SSO does not use token secret
$consumerKey = 'VutaTaro1ktGNXKD';
$consumerSecret = 'S3cr3t P@ssw0rd';          //In UI called "Shared secret"
$signatureMethod = 'HMAC-SHA256';     //or HMAC-SHA1 or PLAINTEXT
$nonce = 'fjaLirsIcCGVZWzBX0pg';      //substr(str_shuffle("0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"), 0,
 20);
$timestamp = '1508242306';            //time();
$version = '1.0';
```

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

```
$baseString = oauth_get_sbs($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

```
GET&https%3A%2F%2F<accountID>.app.netsuite.com%2Fapp%2Fcommon%2Fintegration%2Fssoapplistener.nl&oauth_consumer_key%3DVutaTaro1k
tGNXKD%26oauth_nonce%3DfjaLirsIcCGVZWzBX0pg%26oauth_signature_method%3DHMAC-SHA256%26oauth_timestamp%3D1508242306%26oauth_to
ken%3D030e6a121766126c6b445655477e7252517c395926f3430a%26oauth_version%3D1.0
```

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 about the signature base string.

See also Create the Base String Manually.

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

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

Signature

HMAC-SHA

Signature HMAC-SHA Example

```
$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

```
PP1VMUdgDJeSkeNwJ8EqjKowOVddSWy9JqRT3WQJWck=
```

Signature HMAC-SHA1 Example

```
6nMUbMdr0cssfVDo0YmsBelwnpo=
```

PLAINTEXT

Signature PLAINTEXT

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

Signature PLAINTEXT Example

```
S3cr3t%20P%40ssw0rd&
```

Creating the Authorization Header for SuiteSignOn

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

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

Header

```
$header = 'Authorization: OAuth '
         .'oauth_token="' .rawurlencode($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="VutaTaro1ktGNXKD",
 oauth_nonce="fjaLirsIcCGVZWzBX0pg", oauth_timestamp="1508242306", oauth_signature_method="HMAC-SHA256", oauth_version="1.0",
 oauth_signature="Q6jMu61V%2BORdf6UeZ39ixFSu3rXO2dwwuCq8PlcWNqQ%3D"
```

Header HMAC-SHA1 Example

```
Authorization: OAuth oauth_token="030e6a121766126c6b445655477e7252517c395926f3430a",  oauth_consumer_key="VutaTaro1ktGNXKD",
  oauth_nonce="fjaLirsIcCGVZWzBX0pg", 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="VutaTaro1ktGNXKD", oauth_token="030e6a121766126c6b445655477e7252517c395926f3430a",
 oauth_nonce="fjaLirsIcCGVZWzBX0pg", 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 a screenshot 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.

1. HTTP method - line 3

   Note: The HTTP method must be in uppercase.

```
1  function restletBaseString($httpMethod, $url, $consumerKey, $tokenKey, $nonce, $timestamp, $version, $signatureMethod, $postParams){
2    //http method must be upper case
3    $baseString = strtoupper($httpMethod) .'&';
4
```

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)

```
5    //include url without parameters, schema and hostname must be lower case
6    if (strpos($url, '?')){
7      $baseUrl = substr($url, 0, strpos($url, '?'));
8      $getParams = substr($url, strpos($url, '?') + 1);
9    } else {
10     $baseUrl = $url;
11     $getParams = "";
12   }
13   $hostname = strtolower(substr($baseUrl, 0,  strpos($baseUrl, '/', 10)));
14   $path = substr($baseUrl, strpos($baseUrl, '/', 10));
15   $baseUrl = $hostname . $path;
16   $baseString .= rawurlencode($baseUrl) .'&';
```

3. Parameters - lines 19-51
   - Place 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 in alphabetical order, 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)

```
19    $params = array();
20    $params['oauth_consumer_key'] = array($consumerKey);
21    $params['oauth_token'] = array($tokenKey);
22    $params['oauth_nonce'] = array($nonce);
23    $params['oauth_timestamp'] = array($timestamp);
24    $params['oauth_signature_method'] = array($signatureMethod);
25    $params['oauth_version'] = array($version);
26
27    foreach (explode('&', $getParams ."&". $postParams) as $param) {
28      $parsed = explode('=', $param);
29      if ($parsed[0] != "") {
30        $value = isset($parsed[1]) ? urldecode($parsed[1]): "";
31        if (isset($params[urldecode($parsed[0])])) {
32          array_push($params[urldecode($parsed[0])], $value);
33        } else {
34          $params[urldecode($parsed[0])] = array($value);
35        }
36      }
37    }
38
39    //all parameters must be alphabetically sorted
40    ksort($params);
41
42    $paramString = "";
43    foreach ($params as $key => $valueArray){
44      //all values must be alphabetically sorted
45      sort($valueArray);
46      foreach ($valueArray as $value){
47        $paramString .= rawurlencode($key) . '='. rawurlencode($value) .'&';
48      }
49    }
50    $paramString = substr($paramString, 0, -1);
51    $baseString .= rawurlencode($paramString);
52    return $baseString;
53  }
```

The restletBaseString Function

```
1  function restletBaseString($httpMethod, $url, $consumerKey, $tokenKey, $nonce, $timestamp, $version, $signatureMethod, $postParams)
   {
2    //http method must be upper case
3    $baseString = strtoupper($httpMethod) .'&';
4
5    //include url without parameters, schema and hostname must be lower case
6    if (strpos($url, '?')){
7      $baseUrl = substr($url, 0, strpos($url, '?'));
8      $getParams = substr($url, strpos($url, '?') + 1);
9    } else {
10     $baseUrl = $url;
11     $getParams = "";
12   }
13   $hostname = strtolower(substr($baseUrl, 0,  strpos($baseUrl, '/', 10)));
14   $path = substr($baseUrl, strpos($baseUrl, '/', 10));
15   $baseUrl = $hostname . $path;
```

```
  $baseString .= rawurlencode($baseUrl) .'&';

  //all oauth and get params. First they are decoded, next sorted in alphabetical order, next each key and values is encoded and finally whole parame
ters 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 sorted in alphabetical order
  ksort($params);

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

NetSuite as OIDC Provider

The NetSuite as OIDC Provider feature provides an alternative to NetSuite's Outbound Single Sign-on (SuiteSignOn) feature. This feature is based on OpenID Connect (OIDC) Single Sign-on, with NetSuite acting as the OIDC provider (OP). The NetSuite as OIDC Provider feature uses OAuth 2.0 as an authorization framework. OAuth 2.0 is a token-based authentication method. For more information, see OAuth 2.0.

Note: You should use the NetSuite as OIDC Provider feature as an outbound single sign-on method for use with integrations.

See the following topics for more information about the NetSuite as OIDC Provider feature:

- NetSuite as OIDC Provider Tasks for Administrators
  - Getting Started with NetSuite as OIDC Provider
  - Managing OAuth 2.0 Authorized Applications
- NetSuite as OIDC Provider for Integration Application Developers
- Troubleshooting NetSuite as OIDC Provider

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:

- OpenID Connect (OIDC) Single Sign-on
- SAML Single Sign-on

NetSuite as OIDC Provider Tasks for Administrators

The following topics provide information about tasks that administrators must complete to support the NetSuite as OIDC Provider feature for integrations. See the following topics:

- Getting Started with NetSuite as OIDC Provider
  - Enable the NetSuite as OIDC Provider Feature
  - Add OIDC Provider Setup Permission to Roles

    See also OIDC Provider Setup Permission.
  - Assign Users to Roles with the OIDC Provider Setup Permission
  - Create Integration Records for Applications that Use NetSuite as OIDC Provider for Outbound Single Sign-on
  - Configure NetSuite as OIDC Provider
- Managing OAuth 2.0 Authorized Applications

Getting Started with NetSuite as OIDC Provider

To set up the NetSuite as OIDC Provider feature in your NetSuite account, you must complete the following tasks.

Click the links in the following steps for detailed instructions for each task.

To set up the NetSuite as OIDC Provider feature in your NetSuite account:

1. Enable the NetSuite as OIDC Provider Feature
2. Add OIDC Provider Setup Permission to Roles

   See also OIDC Provider Setup Permission.
3. Assign Users to Roles with the OIDC Provider Setup Permission
4. Create Integration Records for Applications that Use NetSuite as OIDC Provider for Outbound Single Sign-on
5. Configure NetSuite as OIDC Provider
6. Managing OAuth 2.0 Authorized Applications

Enable the NetSuite as OIDC Provider Feature

Before you can begin using the NetSuite as OIDC Provider feature in your account, you must enable the feature.

To enable the NetSuite as OIDC Provider feature:

1. Go to Setup > Company > Setup Tasks > Enable Features.
2. Click the **SuiteCloud** subtab.
3. In the **Manage Authentication** section, check the **OAuth 2.0** box. Click **I Agree** on the SuiteCloud Terms of Service page.

   Note: You must enable the OAuth 2.0 feature to use the NetSuite as OIDC Provider feature in your account.
4. In the **Manage Authentication** section, check the **NetSuite as OIDC Provider** box. Click **I Agree** on the SuiteCloud Terms of Service page.
5. Click **Save**.

After you have enabled the NetSuite as OIDC Provider feature:

- You must set up NetSuite as OIDC Provider roles. See Add OIDC Provider Setup Permission to Roles. See also OIDC Provider Setup Permission.
- Administrators and users with the Integration Application permission can configure applications to use the NetSuite as OIDC Provider feature. See Create Integration Records for Applications that Use NetSuite as OIDC Provider for Outbound Single Sign-on. For more information about integration records, see the help topic Creating an Integration Record.

Add OIDC Provider Setup Permission to Roles

An administrator can create a new role with the OIDC Provider Setup permission, or modify existing roles to add the permission. Then these roles can be assigned to users as needed. For more information about creating or customizing roles, see:

- NetSuite Users & Roles
- NetSuite Roles Overview

OIDC Provider Setup Permission

The following permission can be added to roles as appropriate.

OIDC Provider Setup:

- Enables users to configure the audience for NetSuite as OIDC Provider applications.
- Is primarily for administrators or roles with Core Administration Permissions (CAP). For more information about CAP, see the help topic Core Administration Permissions.
- Requires two-factor authentication (2FA).

Note: The OIDC Provider Setup permission does not allow management of OAuth 2.0 authorized applications in NetSuite accounts. To manage the authorized applications, users need the OAuth 2.0 Authorized Applications Management permission. For more information, see Set Up OAuth 2.0 Roles. See also Managing OAuth 2.0 Authorized Applications.

To add permissions to a role, go to Setup > Users/Roles > User Management > Manage Roles. Select a role to customize. On the Permission tab, Setup subtab, choose the permission from the list and click **Add**.

For more information, see Assign Users to Roles with the OIDC Provider Setup Permission.

Assign Users to Roles with the OIDC Provider Setup Permission

After you have modified roles to add the OIDC Provider Setup permission, you can assign users to these roles. The following is a brief procedure for assigning a role to an existing user. If you need more information about assigning roles to users, see the help topic NetSuite Users Overview.

To assign NetSuite as OIDC Provider roles to users:

1. Go to the entity record for the user:
   - If the user is an employee, go to Lists > Employees > Employees.
2. Click **Edit** next to the name of the user to whom you want to assign the role with the OIDC Provider Setup permission.
3. Click the **Access** tab.
4. On the Roles subtab, in the **Role** field, select the role for this user.
5. Click **Add**.
6. Click **Save**.

Next, you must set up applications to use the NetSuite as OIDC Provider feature for authentication. See Create Integration Records for Applications that Use NetSuite as OIDC Provider for Outbound Single Sign-on.

Create Integration Records for Applications that Use NetSuite as OIDC Provider for Outbound Single Sign-on

Before the code grant flow can be initiated, an integration record must be created for the application. It is also possible to edit an existing integration record to apply to an application that uses NetSuite as OIDC Provider for outbound single sign-on. Administrators and users with the Integration Application permission can create integration records. For more information about integration records, see the help topic Integration Management.

The following procedure describes how to create an integration record.

To create an integration record for an application:

1. Go to Setup > Integration > Integration Management > Manage Integrations > New.

2. Enter a name for your application in the **Name** field.
3. Enter a description in the **Description** field, if preferred.
4. Select Enabled in the **State** field.
5. Enter a note in the **Note** field, if preferred.

> **Note:** Values of the **State** , **Note**, and **OAuth 2.0 Consent Policy** fields are specific to one NetSuite account. If you install a record in a different account, the values may change. Values of the **Name** and **Description** fields are read-only if the record is installed in a different account. For more information, see the help topic Auto-Installation of Integration Records.

6. On the **Authentication** tab, check or clear the appropriate boxes for your application:

Field on the Authentication tab, under OAuth 2.0:	Function of the field:
Authorization Code Grant For more information, see NetSuite as OIDC Provider for Integration Application Developers.	You must check this box for NetSuite as OIDC Provider to work.
Redirect URI	▪ Enter a valid redirect URI, where your application will handle the code. ▪ The redirect URI is validated when you save the integration record. **Important:** The redirect URI must be configured as either the https:// scheme or a custom URL scheme (for example, myapp://callback). The http:// scheme is not supported. The transport layer security must be guaranteed on the redirect URI.
Public Client	(Optional). Check this box if you want to allow OAuth 2.0 public clients with this integration.
Client Credentials (Machine to Machine) Grant	The OAuth 2.0 client credentials flow cannot be used with the NetSuite as OIDC Provider feature.
RESTlets	This scope is only applicable for NetSuite as OIDC Provider integrations that use the access token for OAuth 2.0 authorization. For more information, see OAuth 2.0 for Integration Application Developers.
REST Web Services	This scope is only applicable for NetSuite as OIDC Provider integrations that use the access token for OAuth 2.0 authorization. For more information, see OAuth 2.0 for Integration Application Developers.
SuiteAnalytics Connect	This scope is only applicable for NetSuite as OIDC Provider integrations that use the access token for OAuth 2.0 authorization. For more information, see OAuth 2.0 for Integration Application Developers.
Application Logo	(Optional). You can select a file from your File Cabinet. Supported formats are JPEG, PNG and GIF.
Application Terms of Use	(Optional). You can select any PDF file from your File Cabinet.
Application Privacy Policy	(Optional). You can select any PDF file from your File Cabinet.
OAuth 2.0 Consent Policy	Select an option from the list. See the following for more details about these options: ▪ **Always Ask** - This is the default option. The consent screen appears every time the OAuth 2.0 code grant flow is initiated. ▪ **Never Ask** - The consent screen does not appear during the OAuth 2.0 code grant flow. The integration is autoapproved by an administrator.

	▪ **Ask First Time** - The consent screen only appears the first time the OAuth 2.0 code grant flow is initiated. The consent screen also appears if: ▫ The consent was not given previously ▫ The system does not know which role or account to choose for the user to log in with ▫ The application requires a different set of scopes and needs a new consent Integration application developers can adjust the consent screen option using the prompt parameter in Step One of the OAuth 2.0 code grant flow. For more information, see Step One GET Request to the Authorization Endpoint. See also Integration Record and Prompt Parameter Combinations.

7. Click **Save**

Warning: For security reasons, the only time the client credentials (the client ID and client 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 the client ID and client secret, you will have to reset them on the Integration page to obtain new values. Treat these values like you treat a password. Never share the client ID and client secret with unauthorized individuals and never send them by email.

Next, you must configure the audience to use the integration. For more information, see Configure NetSuite as OIDC Provider.

Configure NetSuite as OIDC Provider

Users in roles with the OIDC Provider Setup permission can configure the audience for NetSuite as OIDC Provider applications, deactivate the applications or reactivate the inactive applications.

NetSuite as OIDC Provider Setup

To configure audience for authorized applications

1. Setup > Integration > Manage Authentication > NetSuite as OIDC Provider Setup
2. Click the link in the **Integration Name** column.
3. In the window, select Entities and Roles you want to enable for this application. You can select all by checking the **Select All** box.

   Note: The **Select All** box selects all current and future Entities and Roles.

   To move the selected Entities or Roles to the right column, click the upper arrow between the columns.

   To clear Entities or Roles, check the box next to their name and click the bottom arrow between the columns.

   You can choose both Entities and Roles at the same time.

   Both entity and role must be enabled for a user to successfully use the NetSuite as OIDC Provider feature.

   Note: In the window, you can also deactivate the application. To do so, check the **Inactive** box.

4. Click **Save**.

Using NetSuite Well–known URI Metadata to configure the Relying Party (RP)

You can find the OIDC configuration metadata on the NetSuite as OIDC Provider Setup page. The metadata file is accessible through the **Metadata URL** link at the top of the page. The metadata file is specific for each account and contains all data needed to complete the setup of the relying party (RP). It is not possible to provide detailed instructions for configuring the relying party, as the configuration steps will vary.

The format of the **Metadata URL** is:

`https://<accountID>.suitetalk.api.netsuite.com/.well-known/openid-configuration`

where <accountID> represents your NetSuite account ID.

Managing OAuth 2.0 Authorized Applications

The NetSuite as OIDC Provider feature uses OAuth 2.0 as an authorization framework. All integration records created for use with the NetSuite as OIDC Provider feature are based on OAuth 2.0. Administrators and users with the OAuth 2.0 Authorized Applications Management permission can view and manage all authorized applications in the account. For more information, see Managing OAuth 2.0 Authorized Applications and Viewing and Revoking OAuth 2.0 Authorized Applications.

NetSuite as OIDC Provider for Integration Application Developers

The NetSuite as OIDC Provider feature is based on the OAuth 2.0 authentication method.

NetSuite as OIDC Provider access is based on the OAuth 2.0 authorization code grant flow for generation of access tokens, refresh tokens, and ID tokens. This alternative is more straightforward than the three-step TBA authorization flow, because it does not require signing of requests.

For more information, see OAuth 2.0 Authorization Code Grant Flow.

OAuth 2.0 Authorization Code Grant Flow

Application developers and integrators can use a redirection-based authorization code grant flow with OAuth 2.0. If there is no active session, users enter user credentials into one of the following login forms as a part of the flow.

- A trusted NetSuite login form
- SAML SSO Identity provider's login form
- OIDC OpenID Connect provider's login form

The OAuth 2.0 authorization code grant flow consists of two steps, an additional refresh token request, and a request to the logout endpoint.

- Step One GET Request to the Authorization Endpoint
- Step Two POST Request to the Token Endpoint
- Refresh Token POST Request to the Token Endpoint
- Request to the Logout Endpoint

With the OAuth 2.0 authorization code grant flow, the application begins the process of granting the access token, refresh token, and ID token by sending a GET request to the authorization endpoint. The user, to whom the access token, refresh token, and ID token are to be granted, explicitly consents to the application accessing NetSuite.

An administrator must create integration records for each application. See Create Integration Records for Applications that Use NetSuite as OIDC Provider for Outbound Single Sign-on. The administrator must configure the redirect URI on the integration record. The underlying application must have the ability to open a browser.

For more information, see RFC 6749.

Step One GET Request to the Authorization Endpoint

In the first step of the OAuth 2.0 authorization code grant flow, the application sends a GET request to the authorization endpoint. This request must include the required parameters in the request header.

The format of the URL is:

`https://<accountID>.app.netsuite.com/app/login/oauth2/authorize.nl`.

where <accountID> represents your NetSuite account ID. If you do not know the specific account ID, requests can be sent to

`https://system.netsuite.com/app/login/oauth2/authorize.nl`.

See the following table for details about parameters for the GET request.

Request Parameters for Step One

Request Parameter	Description
response_type	The value of the response_type parameter is always `code`.
client_id	■ Identifies the client. ■ The value of the client ID is provided when the integration record is created.
redirect_uri	■ The application uses the valid redirect URI to handle the authorization code. ■ The value of the redirect URI parameter must match the redirect URI in the corresponding integration record.
scope	The scope for which the application is requesting access. Values are `openid` and `email`. If the application requests access for both, the values are separated by a white space. **Note:** NetSuite supports three additional scopes that are used for the OAuth 2.0 feature. For more information, see Step One GET Request to the Authorization Endpoint in the OAuth 2.0 topic.
state	The length of the state parameter must be between 24 and 1024 characters. Valid characters are all printable ASCII characters. **Important:** To avoid cross-site request forgery (CSRF) attacks, you must conform to the OAuth 2.0 specification. For more information, see RFC6749 Section 10.12.
code_challenge	■ This parameter is optional; however, you should use this security extension.

Request Parameter	Description
	Important: If you use public clients for NetSuite as OIDC Provider, the code_challenge parameter is required. ■ The code_challenge parameter is created using code_verifier, a random string of characters. For more information, see PKCE specification https://tools.ietf.org/html/rfc7636#section-4.2. ■ Apply SHA256 on the code_verifier parameter. ■ The length of the code_verifier parameter must be between 43 and 128 characters. ■ Valid characters of the code_verifier parameter are alphabet characters, numbers, and non-alpha numeric ASCII characters: hyphen, period, underscore, and tilde (- . _ ~).
code_challenge_method	■ This parameter is optional. However, if you configure the code_challenge parameter, you must configure the code_challenge_method parameter. You must use SHA256 to configure the code_challenge_method parameter. ■ When the authorization server generates an authorization code, the code_challenge and code_challenge_method parameters are associated with the authorization code value, to ensure they are properly verified. For more information, see PKCE specification https://tools.ietf.org/html/rfc7636#section-4.4. **Important:** As of 2020.2, NetSuite does not support a value of `plain` for the code_challenge_method parameter. Use S256 instead.
nonce	The maximum length of the nonce parameter is 256 characters.
prompt	The optional prompt parameter provides additional control of when the consent screen appears. Following are the values you can use with the prompt parameter: ■ `none` - the consent screen does not appear. If there is no active session, the application returns an error. ■ `login` - the user must authenticate even if there is an active session. This option only works if the application sends the request to the account-specific domain. ■ `consent` - the consent screen appears every time. The user must authenticate if there is no active session. ■ `login consent` or `consent login` - the consent screen appears every time, and the user must authenticate even if there is an active session. For more information, see Integration Record and Prompt Parameter Combinations.

Important: Request parameters must be encoded based on the HTML specification for the `application/x-www-form-urlencoded` media type. For more information, see URL Specification 5.1.

The following URL provides a sample GET request.

```
https://<accountID>.app.netsuite.com/app/login/oauth2/authorize.nl?scope=openid+email&redirect_uri=https%3A%2F%2Fmyapplication.com%2Fnetsuite%2Foauth2callback&response_type=code&client_id=6794a3086e4f61a120350d01b8527aed3631472ef334122124 95be65a8fc8d4c&state=ykv2XLx1BpT5Q0F3MRPHb94j&nonce=ym1W7YfRHweT46AcCX8MFajz&code_challenge=Who5QBshz2Mu1Mq6GuAknYA5TnjA-0z7VhAgLloec1s&code_challenge_method=S256
```

Consent Screen

After the application sends the GET request, the system displays the consent screen, where a user authorizes the application to access NetSuite through RESTlets or REST web services.

Important: If there is no active NetSuite session, the user is first redirected to the NetSuite login form. If the GET request points to an account-specific domain, for an account with SAML SSO or OIDC enabled, the user may be redirected to a third party application. After successful authentication, the system displays the consent screen.

The consent screen includes the following:

- The **Application Logo**, **Terms of Use**, and **Privacy Policy**, if these values were entered in the integration record.
- A data and a role to which the application requests access.
- The **Allow/Continue** button. If the application was not previously authorized, the user must click **Allow** to authorize the application. If the application was previously authorized, the user must click a **Continue** button to continue to the next step of the flow.
- The **Deny/Go Back** button. If the application was not previously authorized, the user can click **Deny** to interrupt the flow. If the application was previously authorized, the user can click **Go Back** to interrupt the flow.
- The **Choose Another Role** list. The user can change a role that authorizes the application, by clicking the **Choose Another Role** link.

Redirect Parameters for Step One

After authorization, NetSuite initiates a redirect to the Redirect URI, with the following parameters:

Redirect Parameter	Description
state	The state parameter in the redirect must match the state parameter in the request in Step One. To avoid cross-site request forgery (CSRF) attacks, you must conform to the OAuth 2.0 specification. For more information, see RFC6749 Section 10.12.
code	■ A randomly generated string that is used for request verification in Step Two. ■ The code parameter is only generated if the application was authorized. ■ You must use the value of the code parameter immediately after it is generated. The value for the code parameter has limited time validity.
role	Indicates the user's role for which the access token and refresh token are granted in Step Two. The role parameter is a NetSuite-specific parameter.
entity	The ID of the user who authorizes the application or interrupts the flow. The entity parameter is a NetSuite-specific parameter.
company	NetSuite account ID (company identifier). The company parameter is a NetSuite-specific parameter.
error	The error parameter is only used when an error occurs during the flow. For information about error values, see Troubleshooting NetSuite as OIDC Provider.

The following sample redirects illustrate successful and unsuccessful authorization.

- Application successfully authorized:

```
https://myapplication.com/netsuite/oauth2callback?state=ykv2XLx1BpT5Q0F3MRPHb94j&role=1000&entity=12&company=1234567&code=70b827f926a512f098b1289f0991abe3c767947a43498c2e2f80ed5aef6a5c50
```

- Application not authorized:

```
https://myapplication.com/netsuite/oauth2callback?state=ykv2XLx1BpT5Q0F3MRPHb94j&role=1000&entity=12&company=1234567&error=access_denied
```

After the request to the redirect URI is sent, the flow proceeds to Step Two POST Request to the Token Endpoint.

Step Two POST Request to the Token Endpoint

The application sends a POST request to the token endpoint. The request must include client credentials in the HTTP authorization request header and the required parameters in the request body. At the end of this step, the access token, refresh token, and ID token are granted.

Note: If you use public clients you can choose from the following options:

- The HTTP authorization request header does not contain the Authorization. Additionally, the PKCE parameters and the client_id parameter are included in the body of the request, or
- The HTTP authorization request header contains only the client_id in the Authorization. The PKCE parameters are included in the body of the request.

The format of the URL is:

`https://<accountID>.suitetalk.api.netsuite.com/services/rest/auth/oauth2/v1/token`

where <accountID> is your NetSuite account ID.

Request Parameters for Step Two

Request Parameter	Description
code	The code parameter value obtained in Step One.
redirect_uri	The value of the redirect_uri parameter must match the value entered in the corresponding integration record and the value in the request in Step One.
grant_type	The value of the grant_type parameter in Step Two is `authorization_code`.
code_verifier	If the value of the code_verifier parameter does not match the value generated in Step One, an HTTP 400 Bad Response error is returned. For more information, see https://tools.ietf.org/html/rfc7636, sections 4.5 and 4.6.

Important: Be aware of the following requirements for the request:

- Request parameters must be encoded based on the HTML specification for the `application/x-www-form-urlencoded` media type. For more information, see URL Specification 5.1
- The client authentication method used in the header of the request follows the HTTP Basic authentication scheme. For more information, see RFC 7617. The format is **clientid:clientsecret**. The string value is Base64 encoded. The following code provides an example.

```
POST /services/rest/auth/oauth2/v1/token HTTP/1.1
```

```
Host: <accountID>.suitetalk.api.netsuite
Authorization: Basic Njc5NGEzMDg2ZTRmNjFhMTIwMzUwZDAxYjg1MjdhZWQzNjMxNDcyZWYzMzQxMjIxMjQ5NWJlNjVhOGZjOGQ0YzpjZGM3YWMyMjE4M2VmNTAyNGU4MWIwZmNlOGVmNDYxYzQ0ZDU4OTZhMWYxODA1ZDRiMzcyY2E2MWM0ZDMyNmFl
Content-Type: application/x-www-form-urlencoded

code=70b827f926a512f098b1289f0991abe3c767947a43498c2e2f80ed5aef6a5c50&redirect_uri=https%3A%2F%2Fmyapplication.com%2Fnetsuite%2Foauth2callback&grant_type=authorization_code&code_verifier=abFOm_isZAwm7PpI9BtJRMEuiMqhU6sUqZlVWSsAAf1QutgIOD~on78mu-JdpbKc_RA7IEcf2e~q0XrKlJ1tE.8Un64PXLKQG16G4lwW-a5de_0aeU2mHnyVPg.Or8cE
```

Note: If you use public clients you can choose from the following options:

- The HTTP authorization request header does not contain the Authorization. Additionally, the PKCE parameters and the client_id parameter are included in the body of the request, or
- The HTTP authorization request header contains only the client_id in the Authorization. The PKCE parameters are included in the body of the request.

The following code provides an example of the HTTP authorization request with the PKCE parameters and the client_id parameter included in the body of the request:

```
POST /services/rest/auth/oauth2/v1/token HTTP/1.1
Host: <accountID>.suitetalk.api.netsuite
Content-Type: application/x-www-form-urlencoded

code=70b827f926a512f098b1289f0991abe3c767947a43498c2e2f80ed5aef6a5c50&redirect_uri=https%3A%2F%2Fmyapplication.com%2Fnetsuite%2Foauth2callback&grant_type=authorization_code%client_id=6794a3086e4f61a120350d01b8527aed3631472ef33412212495be65a8fc8d4c&code_verifier=XG2JcZ.I5_67es~Pev0ZbWSPlOklZJq-KPNaFInKuzgGQV4AWt5taWLDnD4IAsnJW_hl9iPQdQcv9~xGSY.qTiB99HA2rfm8cUwlfrzBY0j3bK4XPx-gLhoV1MF1JCC2
```

The following code provides an example of the HTTP authorization request with the client_id parameter included in the Authorization. The PKCE parameters are included in the body of the request:

```
POST /services/rest/auth/oauth2/v1/token HTTP/1.1
Host: <accountID>.suitetalk.api.netsuite
Authorization: Basic Njc5NGEzMDg2ZTRmNjFhMTIwMzUwZDAxYjg1MjdhZWQzNjMxNDcyZWYzMzQxMjIxMjQ5NWJlNjVhOGZjOGQ0Yzo=
Content-Type: application/x-www-form-urlencoded

code=70b827f926a512f098b1289f0991abe3c767947a43498c2e2f80ed5aef6a5c50&redirect_uri=https%3A%2F%2Fmyapplication.com%2Fnetsuite%2Foauth2callback&grant_type=authorization_code%code_verifier=XG2JcZ.I5_67es~Pev0ZbWSPlOklZJq-KPNaFInKuzgGQV4AWt5taWLDnD4IAsnJW_hl9iPQdQcv9~xGSY.qTiB99HA2rfm8cUwlfrzBY0j3bK4XPx-gLhoV1MF1JCC2
```

HTTP Response for Step Two

JSON Response Fields	Description
access_token	The value of the access_token parameter is in JSON Web Token (JWT) format. The access token is valid for 60 minutes.
refresh_token	The value of the refresh_token parameter is in JSON JWT format. The refresh token is valid for seven days. **Important:** If you use public clients for OAuth 2.0, the refresh token is only valid for three hours and is for one-time use only.
expires_in	The value of the expires_in parameter is always 3600. The value represents the time period during which the access token is valid, in seconds.

JSON Response Fields	Description
token_type	The value of the type parameter is always `bearer`.
id_token	The value of the id_token parameter is in JSON JWT format. The id token is valid for three hours.

The following is an example of a response in JSON JWT format:


```
{"access_token":"eyJraWQiOiJzLlNZU1RFTS4yMDIwXzEiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiI1NTstNSIsImF1ZCI6WyJEREUyN
DFGRC1ENjVGLTQ0REQtQUNBRC0wRjJEQ0MzRDhCRkM7TVNUUldMRkNBTkFEQSIsIjk5MDFjZTQxZTAxZjdlZDlkZmY4NDliNGRjYTVmNTVlZDIyYjliNmY5M
mIyY2ViZTViMTUxZjhiYTM5NzQ0MTUiXSwic2NvcGUiOlsiZW1haWwiLCJvcGVuaWQiXSwiaXNzIjoiaHR0cHM6XC9cL3N5c3RlbS5uZXRzdWl0ZS5jb20iLCJvaX
QiOjE2MTMwNTQ0MDksImV4cCI6MTYxMzA1ODAwOSwiaWF0IjoxNjEzMDU0NDA5LCJqdGkiOiJNU1RSV0xGQ0FOQURBLmEuODY2MTczOTUtZDlmYi00OGIyLWIyY
mUtMzlhZWQyODQ3YjhhXzE2MTMwNTQ0MDk1OTMuMTYxMzA1NDQwOTU5MyJ9.kOaxEPGrDjiNU2Zn2IxQbRk132W2YcUPu9vusBD64XpKhk00nDajOvcLSHMyC0k7hd
NPhHr2MkzhzFkHTJzA9SA6czAgTq34NAknYYTFdxdbVp4A8PFyXwcXke6Plu-YIzFkPwSGquMBU1eW93YmcjU6V4Svp5Nz90sog2AV74xr_hjruSxlLAJ2uIuxMhFrnJsx
mUYKqxhzKUkaIIvAf_nbweLNdfx5Vms1LeZZEAIRRoqp1ugpTbVUKLbJwVhw9s8K_NW7WcTIIuIrK3Sukstk6M-tvm04DpyU3SJMriKuWZtTI9J4e1yI0BfyJ4DB4E_L6Ay
dWrVefM7WYRD_TQ","refresh_token":"eyJraWQiOiJzLlNZU1RFTS4yMDIwXzEiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiI1NTstNSIsIm
F1ZCI6WyJEREUyNDFGRC1ENjVGLTQ0REQtQUNBRC0wRjJEQ0MzRDhCRkM7TVNUUldMRkNBTkFEQSIsIjk5MDFjZTQxZTAxZjdlZDlkZmY4NDliNGRjYTVmNTVlZDIyYjliN
mY5MmIyY2ViZTViMTUxZjhiYTM5NzQ0MTUiXSwic2NvcGUiOlsiZW1haWwiLCJvcGVuaWQiXSwiaXNzIjoiaHR0cHM6XC9cL3N5c3RlbS5uZXRzdWl0ZS5jb20iLCJvaX
QiOjE2MTMwNTQ0MDksImV4cCI6MTYxMzY1OTIwOSwiaWF0IjoxNjEzMDU0NDA5LCJqdGkiOiJNU1RSV0xGQ0FOQURBLnIuODY2MTczOTUtZDlmYi00OGIyLWIyYmUtMzl
hZWQyODQ3YjhhXzE2MTMwNTQ0MDk1OTMuMCJ9.R20NH4bOSZFY8fF_BQasT-_H9vdlLPQUUjT_vk-QhldCp0ldtKDuYk9jZQyyuFhvDeJymRfgnGcG9FGoEz3ArN7Vrox
ZiPlHjG-lk7TDn0WeUR6Gqsvf9ITLfrUj5vy-Nqux91WesE02g6WFRt8YOz1o0D4wagmfpHLIgNpLmMGzpDwOzgOxvUBwLwXrVMptj_gPKrsIFafuak7my8xy332f3ZW
bG80VSra_wif4Kgy1NGJ8BCLcy6t5Q4RgJ1pgEau52rkBi6EqcNyPsFvzpkElmQq8S9yJo9ALXrNIXrx6XP-kDAR7Pai3lpqZGEYVi-DODYBDimgj-bWG1HPT3w","id_to
ken": "eyJraWQiOiJjLjQwMzAwNTkuMjAyMS0wMS0wNV8yMy00Ny0yOSIsInR5cCI6IkpXVCIsImFsZyI6IlJTMjU2In0.eyJhdF9oYXNoIjoiZWYtTXAxaHhVNkdr
cjdIcFFwd0xJeEhJOHo0TGR5bG1ac0IxWGdiNVJYT1VBMmhPU3hydHlvdFNpc0gtamRNS25SV3hXT3FRdWxhd2tzM2JzdHpyMEtERGl5Tl80Sk9fNVdCTkNBVTFIM
llJaFVQdmkyYUFVcGxSbDZPSnpPQmN2cUItSmNYcXJaVk94aU01eXBnMWtDRTlGeWxzOWFwV3pBZE04Q2pVVnVUMG9HLWs0eFM3cTBFMWRwYlZ0WXZwXzVEZDFfTzdL
VEtvc1B3SENKRzhHWRnUVR2S0NKVWpBSlVVZWxDVFVBdm5OMnAyZEF6ZkVRWmdxVng0VVBIUFVUUkhTVVF0bFltZFgtNnI4dFdIZ1JIcXpMbzg0cjZvc214WnM4Tk45MH
lyWkpyLVNUYm9KajJ0dFpmdjNTVDk5ZW1yVEgwUXhKZXgyYm5qOXlrTl9NckY4UGRWOXFSc0VoQVJxSE1Xa1RUWHpVNkxDLTg1Ry1HbUVKeFBraEd2RjBZbFpLVzFBRkR
wa0VZbTRJb29CcEtNVmU3N0ZLZWkwY1lpc2xyVHNNdE1JOGNEeUVzSWVkZlV6SVNFcEVnRDREOVo0X1JPWEhQYzZabFZJVmpVNmlOSEJMTC1uTVNMU0I2QWk0dUxkMDdT
bk5IQW9OWDk5QV9lZmdXaVpuQXZOeWwwdEpHZGpONWMwOGVWV3BCVFRVTDNfNVhOcy1BOUx5czlRV3cwbi1XNVRFMnhoamtNcS1NUFgwN3hRWGdiSUR3N04zMHE0YXljd
mZsMnZicVl6Q0I4dGdzUWtnMUJQNTlKajNDaVZNUWlSeUJod1I0eF9kb3pnNEJhdEZIalA4M3M2eUtuYWx6UzByTU1lbkkta1BweWJwQ3hpZDFFTF9HSHkwN3cxVUUiL
CJzdWIiOiIzOzciLCJhdWQiOlsiMjc5NEM3Q0QtMTRGRC00NTI0LThGNUYtMzVBRTI1MjMzOTM2OzQwMzAwNTkiLCI2NjExMzFmN2JmMGEyZjhhNGQwOWM3OWQzZmE5N
jFlYWI2NjEwMmRjYTQzZTA3YWQ0N2QzYTI5NjI4Y2VkNjdiIl0sImVtYWlsX3ZlcmlmaWVkIjpmYWxzZSwic2NvcGUiOlsiZW1haWwiLCJyZXN0X3dlYnNlcnZpY2VzIi
wib3BlbmlkIiwicmVzdGxldHMiXSwiaXNzIjoiaHR0cHM6XC9cL3N5c3RlbS5uZXRzdWl0ZS5jb20iLCJleHAiOjE2MDk5MzA5MzYsImlhdCI6MTYwOTkyMDEzNi
wibm9uY2UiOiJ5akp6U1J4Wm50NTFtTWlUbHBrYkEwcW8iLCJqdGkiOiI0MDMwMDU5LmkuMDdkMzIwNmQtMjBjNy00Zjk2LWI1OTQtZDVjM2Y2OGZjYjQ2XzE2MDk5M
jAxMzY2MzAuMCIsImVtYWlsIjoibmxidWlsZEbuZXRzdWl0ZS5jb20ifQ.UxcZvby2AQ1a9_037bcAXw66LzkM4MDbrIDrBd85Q2yo2_gpKrCh5jP7jTyM_H9AVTp6hnvb
FilQe_487HW-bLJXJugvvG_Z8StBt-sfbpe-TnxukjyN_-mXCueGoe-Fs6BrtdDy8oQA9jsYmDmKKOEKpb4DexN3S_ADG4oitTZ9ZRccmNe5H8ChijqASIRIcNBTDv0jA
HoFODgSigvGvu7OVeuM2J5bMKohayQoXtHQGMuZov9Tg4_iUh2mcoXxILLAZXiK8K7lxY5__qIgq4Jr1Oayuf9GnD6aHqdFMVMLGcMMu2_N-mVOlqoE7yEx-C3qQ1GFlM
RikZ2s2rEtbCNLSp_Gn0HgEA8iRfkBrBYPWWaSM4M4pDdyszzkB4g-bdSzM10oKcDDDj1SII7cHx71b_ZtD1UCdOQIF8cYjVZEdOMGjqPLEjUGlOh-R1IgRqyxS6Tg
WSS0u2Wyn9EVJJI8MnHbEb9GjVu9QAsza-rgNiLzfFPq9l3fRW5qvn5IkPblOfReN7b2f16tbUzPpzs2l0-NQh4Wfetydbr3VZDAoFrfH3F-1JEe0-PHaCxS5SJq_vH
ErnB1Cz8mU7lpAyr83wVjKCcS2qKCN1FdRm4iJusZ2umxoNX5-VUx0bsOiKnLDoEHEkzYKtqBCs-ARYsyfF5zcSSQddrNhqFPmtg","expires_in":"3600","to
ken_type":"bearer"}
```


Note: The access token, refresh token, and id token are Base64 encoded. For more information, see RFC 7519.

After the tokens are granted, the integration record is auto-installed in the account. If the integration record fails to auto-install, check the **State** field in the corresponding integration record. Go to Setup > Integration > Manage Integrations, and click the name of the corresponding integration record. The value of the **State** field must be **Enabled** to auto-install the integration record successfully.

Note: The integration record should be auto-installed when the access token is granted, if the **Require Approval During Auto-installation of Integration** box is not checked on the SOAP Web Services Preferences page. If this box is checked, you must clear it to successfully auto-install integrations.

Refresh Token POST Request to the Token Endpoint

When the access token expires, the application can send the refresh token POST request to the token endpoint to get a new access token.

The format of the URL is:

`https://<accountID>.suitetalk.api.netsuite.com/services/rest/auth/oauth2/v1/token`

where <accountID> represents your NetSuite account ID.

Request Parameters for the Refresh Token Request

Request Parameter	Description
grant_type	The value of the grant_type parameter is `refresh_token`.
refresh_token	The value of the refresh_token parameter is in JSON Web Token (JWT) format.

Important: the client authentication method used in the header of the request follows the HTTP Basic authentication scheme. For more information, see RFC 7617. The format is **client_id:client_secret**. The string value is Base64 encoded. The following code provides an example.

```
POST /services/rest/auth/oauth2/v1/token HTTP/1.1
Host: <accountID>.suitetalk.api.netsuite.com
Authorization: Basic Njc5NGEzMDg2ZTRmNjFhMTIwMzUwZDAxYjg1MjdhZWQzNjMxNDcyZWYzMzQxMjIxMjQ5NWJlNjVhOGZjOGQ0YzpjZGM3YWMyMjE4M2VmNTAyNGU4MWIwZmNlOGVmNDYxYzQ0ZDU4OTZhMWYxODA1ZDRiMzcyY2E2MWM0ZDMyNmFl
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token&refresh_token=eyJraWQiOiJzLlNZU1RFTS4yMDIwXzEiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiI1NTstNSIsImF1ZCI6WyJEREUyNDFGRC1ENjVGLTQ0REQtQUNBRC0wRjJEQ0MzRDhCRkM7TVNUUldMRkNBTkFEQSIsIjk5MDFjZTQxZTAxZjdlZDlkZmY4NDliNGRjYTVmNTVlZDIyYjliNmY5MmIyY2ViZTViMTUxZjhiYTM5NzQ0MTUiXSwic2NvcGUiOlsiZW1haWwiLCJvcGVuaWQiXSwiaXNzIjoiaHR0cHM6XC9cL3N5c3RlbS5uZXRzdWl0ZS5jb20iLCJvaXQiOjE2MTMwNTQ0MDksImV4cCI6MTYxMzY1OTIwOSwiaWF0IjoxNjEzMDU0NDA5LCJqdGkiOiJNU1RSV0xGQ0FOQURBLnIuODY2MTczOTUtZDlmYi00OGIyLWIyYmUtMzlhZWQyODQ3YjhhXzE2MTMwNTQ0MDk1OTMuMCJ9.R20NH4bOSZFY8fF_BQasT-_H9vdlLPQUUjT_vk-QhldCp0ldtKDuYk9jZQyyuFhvDeJymRfgnGcG9FGoEz3ArN7VroxZiPlHjG-lk7TDn0WeUR6Gqsvf9ITLfrUj5vy-Nqux91WesE02g6WFRt8YOz1o0D4wagmfpHLIgNpLmMGzpDwOzgOxvUBwLwXrVMptj_gPKrsIFafuak7my8xy332f3ZWbG80VSra_wif4Kgy1NGJ8BCLcy6t5Q4RgJ1pgEau52rkBi6EqcNyPsFvzpkElmQq8S9yJo9ALXrNIXrx6XP-kDAR7Pai3lpqZGEYVi-DODYBDimgj-bWG1HPT3w
```

Note: If you use public clients you can choose from the following options:

- The HTTP authorization request header does not contain the Authorization and the client_id parameter is included in the body of the request, or
- The HTTP authorization request header contains only the client_id in the Authorization.

HTTP Response for Refresh Token Request

JSON Response Fields	Description
access_token	The value of the access_token parameter is in JSON Web Token (JWT) format. The access token is valid for 60 minutes.
expires_in	The value of expires_in parameter is always `3600`. The value represents the time period during which the access token is valid, in seconds.
token_type	The value of the token_type parameter is always `bearer`.

Important: If you use public clients with OAuth 2.0, the refresh token request returns access and refresh token. The refresh token is valid for three hours and is for one-time use only.

The following is an example of a response:

```
{"access_token":"eyJraWQiOiJzLlNZU1RFTS4yMDIwXzEiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiI1NTstNSIsImF1ZCI6WyJEREUyN
DFGRC1ENjVGLTQ0REQtQUNBRC0wRjJEQ0MzRDhCRkM7TVNUUldMRkNBTkFEQSIsIjk5MDFjZTQxZTAxZjdlZDlkZmY4NDliNGRjYTVmNTVlZDIyYjliNmY5M
mIyY2ViZTViMTUxZjhiYTM5NzQ0MTUiXSwic2NvcGUiOlsiZW1haWwiLCJvcGVuaWQiXSwiaXNzIjoiaHR0cHM6XC9cL3N5c3RlbS5uZXRzdWl0ZS5jb20iLCJvaX
QiOjE2MTMwNTQ0MDksImV4cCI6MTYxMzA1ODAwOSwiaWF0IjoxNjEzMDU0NDA5LCJqdGkiOiJNU1RSV0xGQ0FOQURBLmEuODY2MTczOTUtZDlmYi00OGIyLWIyY
mUtMzlhZWQyODQ3YjhhXzE2MTMwNTQ0MDk1OTMuMTYxMzA1NDQwOTU5MyJ9.kOaxEPGrDjiNU2Zn2IxQbRk132W2YcUPu9vusBD64XpKhk00nDajOvcLSHMyC0k7hd
NPhHr2MkzhzFkHTJzA9SA6czAgTq34NAknYYTFdxdbVp4A8PFyXwcXke6Plu-YIzFkPwSGquMBU1eW93YmcjU6V4Svp5Nz90sog2AV74xr_hjruSxlLAJ2uIuxMhFrnJsx
mUYKqxhzKUkaIIvAf_nbweLNdfx5Vms1LeZZEAIRRoqp1ugpTbVUKLbJwVhw9s8K_NW7WcTIIuIrK3Sukstk6M-tvm04DpyU3SJMriKuWZtTI9J4e1yI0BfyJ4DB4E_L6Ay
dWrVefM7WYRD_TQ","expires_in":"3600","token_type":"bearer"}
```


Note: The access token is Base64 encoded. For more information, see RFC 6749. section 1.4.

When the refresh token expires, the token endpoint returns an invalid_grant error. The application must go back to Step One of the OAuth 2.0 authorization code grant flow to restart the process.

Request to the Logout Endpoint

The application sends a GET or POST request to the logout endpoint to revoke the valid id token and its associated access and refresh tokens. This endpoint has the following format:

`https://<accountID>.suitetalk.api.netsuite.com/services/rest/auth/oauth2/v1/logout`

where <accountID> represents your NetSuite account ID.

Request Parameters for the Logout Token Request

Request Parameter	Description
id_token_hint	The value of the id_token_hint parameter is the value of the id token that the application revokes.

Important: The client authentication method used in the header of the request follows the HTTP Basic authentication scheme. For more information, see RFC 7617. The format is **client_id:client_secret**. The string value is Base64 encoded. The following code provides an example.

The client authentication in the request to the logout endpoint is optional.

```
POST /services/rest/auth/oauth2/v1/logout HTTP/1.1
Host: <accountID>.suitetalk.api.netsuite.com
Authorization: Basic Njc5NGEzMDg2ZTRmNjFhMTIwMzUwZDAxYjg1MjdhZWQzNjMxNDcyZWYzMzQxMjIxMjQ5NWJlNjVhOGZjOGQ0YzpjZGM3YWMyMjE4M2VmNTAyN
GU4MWIwZmNlOGVmNDYxYzQ0ZDU4OTZhMWYxODA1ZDRiMzcyY2E2MWM0ZDMyNmFl
Content-Type: application/x-www-form-urlencoded

id_token_hint=eyJraWQiOiJjLjQwMzAwNTkuMjAyMS0wMS0wNV8yMy00Ny0yOSIsInR5cCI6IkpXVCIsImFsZyI6IlJTMjU2In0.eyJhdF9oYXNoIjoiZWYtTXAxaH
hVNkdrcjdIcFFwd0xJeEhJOHo0TGR5bG1ac0IxWGdiNVJYT1VBMmhPU3hydHlvdFNpc0gtamRNS25SV3hXT3FRdWxhd2tzM2JzdHpyMEtERGl5Tl80Sk9fNVdCTkNBVT
FIMllJaFVQdmkyYUFVcGxSbDZPSnpPQmN2cUItSmNYcXJaVk94aU01eXBnMWtDRTlGeWxzOWFwV3pBZE04Q2pVVnVUMG9HLWs0eFM3cTBFMWRwYlZ0WXZwXzVEZDFfTzdL
VEtvc1B3SENKRzhHWWRnUVR2S0NKVWpBSlVVZWxDVFVBdm5OMnAyZEF6ZkVRWmdxVng0VVBIUFVUUkhTVVF0bFltZFgtNnI4dFdIZ1JIcXpMbzg0cjZvc214WnM4Tk45MH
lyWkpyLVNUYm9KajJ0dFpmdjNTVDk5ZW1yVEgwUXhKZXgyYm5qOX1rTl9NckY4UGRWOXFSc0VoQVJxSE1Xa1RUWHpVNkxDLTg1Ry1HbUVKeFBraEd2RjBZbFpLVzFBRkR
wa0VZbTRJb29CcEtNVmU3N0ZLZWkwY1lpc2xyVHNNdE1JOGNEeUVzSWVkZlV6SVNFcEVnRDREOVo0X1JPWEhQYzZabFZJVmpVNmlOSEJMTC1uTVNMU0I2Qwk0dUxkMDdT
bk5IQW9OWDk5QV9lZmdXaVpuQXZOeWwwdEpHZGpONWMwOGVWV3BCVFRVTDNfNVhOcy1BOUx5czlRV3cwbi1XNVRFMnhoamtNcS1NUFgwN3hRWGdiSUR3N04zMHE0YXljd
mZsMnZicVl6Q0I4dGdzUWtnMUJQNTlKajNDaVZNUWlSeUJod1I0eF9kb3pnNEJhdEZIalA4M3M2eUtuYWx6UzByTU1lbkkta1BweWJwQ3hpZDFFTF9HSHkwN3cxVUUiL
CJzdWIiOiIzOzciLCJhdWQiOlsiMjc5NEM3Q0QtMTRGRC00NTI0LThGNUYtMzVBRTI1MjMzOTM2OzQwMzAwNTkiLCI2NjExMzFmN2JmMGEyZjhhNGQwOWM3OWQzZmE5N
jFlYWI2NjEwMmRjYTQzZTA3YWQ0N2QzYTI5NjI4Y2VkNjdiIl0sImVtYWlsX3ZlcmlmaWVkIjpmYWxzZSwic2NvcGUiOlsiZW1haWwiLCJyZXN0X3dlYnNlcnZpY2VzIi
wib3BlbmlkIiwicmVzdGxldHMiXSwiaXNzIjoiaHR0cHM6XC9cL3N5c3RlbS5uZXRzdWl0ZS5jb20iLCJleHAiOjE2MDk5MzA5MzYsImlhdCI6MTYwOTkyMDEzNi
wibm9uY2UiOiJ5akp6U1J4Wm50NTFtTWlUbHBrYkEwcW8iLCJqdGkiOiI0MDMwMDU5LmkuMDdkMzIwNmQtMjBjNy00Zjk2LWI1OTQtZDVjM2Y2OGZjYjQ2XzE2MDk5M
jAxMzY2MzAuMCIsImVtYWlsIjoibmxidWlsZEBuZXRzdWl0ZS5jb20ifQ.UxcZvby2AQ1a9_037bcAXw66LzkM4MDbrIDrBd85Q2yo2_gpKrCh5jP7jTyM_H9AVTp6hnvb
FilQe_487HW-bLJXJugvvG_Z8StBt-sfbpe-TnxukjyN_-mXCueGoe-Fs6BrtdDy8oQA9jsYmDmKKOEKpb4DexN3S_ADG4oitTZ9ZRccmNe5H8ChijqASIRIcNBTDv0jA
HoFODgSigvGvu7OVeuM2J5bMKohayQoXtHQGMuZov9Tg4_iUh2mcoXxILLAZXiK8K7lxY5__qIgq4Jr1Oayuf9GnD6aHqdFMVMLGcMMu2_N-mVOlqoE7yEx-C3qQ1GFlM
RikZ2s2rEtbCNLSp_Gn0HgEA8iRfkBrBYPWWaSM4M4pDdyszzkB4g-bdSzM10oKcDDDj1SII7cHx71b_ZtD1UCdOQIF8cYjVZEdOMGjqPLEjUGlOh-R1IgRqyxS6Tg
```

```
WSS0u2Wyn9EVJJI8MnHbEb9GjVu9QAsza-rgNiLzfFPq9l3fRW5qvn5IkPblOfReN7b2f16tbUzPpzs2l0-NQh4WfetydbR3VZDAoFrfH3F-1JEe0-PHaCxS5SJq_vHEr
B1Cz8mU7lpAyr83wVjKCcS2qKCN1FdRm4iJusZ2umxoNX5-VUx0bsOiKnLDoEHEkzYKtqBCs-ARYsyfF5zcSSQddrNhqFPmtg
```

Note: If you use public clients you can choose from the following options:

- The HTTP authorization request header does not contain the Authorization and the client_id parameter is included in the body of the request, or
- The HTTP authorization request header contains only the client_id in the Authorization.

The request to the logout endpoint also invalidates active session of NetSuite.

Integration Record and Prompt Parameter Combinations

See the following table for details about possible combinations of the **Consent Policy** list on the integration record and the prompt parameter in Step One of the OAuth 2.0 code grant flow.

The Consent Policy list value	The prompt parameter value	The Consent Screen
Always Ask	—	The consent screen appears.
Always Ask	none	The consent screen appears.
Always Ask	login	The consent screen appears. A user must authenticate even if there is an active session.
Always Ask	consent	The consent screen appears.
Ask First Time	—	The consent screen appears for the first time. Consent screen also appears if: ■ The consent was not given previously ■ The system does not know which role or account to choose for the user to log in with ■ The application requires a different set of scopes and needs a new consent
Ask First Time	none	The consent screen appears for the first time.
Ask First Time	login	The consent screen appears for the first time. Consent screen also appears if: ■ The consent was not given previously ■ The system does not know which role or account to choose for the user to log in with ■ The application requires a different set of scopes and needs a new consent A user must authenticate even if there is an active session.
Ask First Time	consent	The consent screen appears.
Never Ask	—	The consent screen does not appear.
Never Ask	none	The consent screen does not appear.

The Consent Policy list value	The prompt parameter value	The Consent Screen
Never Ask	login	The consent screen does not appear. A user must authenticate even if there is an active session.
Never Ask	consent	The consent screen appears.

For more information, see Step One GET Request to the Authorization Endpoint and Create Integration Records for Applications that Use NetSuite as OIDC Provider for Outbound Single Sign-on

OAuth 2.0 Token Structure and Certificate Rotation

See the following sections for information about the access, refresh, and id token structure, and the certificate rotation:

- Access and Refresh Token Structure
- ID Token Structure
- Certificate Rotation

Access and Refresh Token Structure

Both the access token and refresh token include three parts: a header, a payload, and a signature.

The token header includes the following parameters:

Parameter Name	Description
kid	The value of the kid parameter is the ID of the certificate used for signing the token.
typ	The value of the type parameter is `JWT`.
alg	The value of the alg parameter is `RS256`.

The token payload includes the following parameters:

Parameter Name	Description
sub	The value of the sub parameter is the role and entity of the user, separated by a semicolon. For example, 1111;10.
aud	The value of the aud parameter is the integration record and the company, separated by a semicolon. Additionally, the client ID is a part of the aud parameter, separated by a comma. For example, 1A111AA1–AA11–1A11–1111–A1A1111111A1;1111, 661131f7bf0a2f8a4d09c79d3fa961eab66102dca43e07ad47d3a29628ced67b.
scope	The value of the scope parameter is either `openid`, `email`, or both, separated by a comma.
iss	The value of the iss parameter is `https://system.netsuite.com`.
oit	The value of the oit parameter represents the number of seconds since the first token of the token chain was issued. This is only applicable for public clients.

Parameter Name	Description
exp	The value of the exp parameter represents the number of seconds since January 1, 1970, until the token's expiration.
iat	The value of the iat parameter represents when the token was issued. The value of the parameter is in seconds, since January 1, 1970.
jti	The value of the jti parameter is the token ID, which is unique for every token.

Note: The token's dot-separated values are Base64 encoded.

The signature is validated with a public key associated with the kid parameter. To access public keys for your account, use the following URL:

`https://<accountID>.suitetalk.api.netsuite.com/services/rest/auth/oauth2/v1/keys`

where <accountID> represents your NetSuite account ID.

ID Token Structure

The id token includes three parts: a header, a payload, and a signature.

The token header includes the following parameters:

Parameter Name	Description
kid	The value of the kid parameter is the ID of the certificate used for signing the token.
typ	The value of the type parameter is `JWT`.
alg	The value of the alg parameter is `RS256`.

The token payload includes the following parameters:

Parameter Name	Description
at_hash	The value of the at_hash parameter is the signature of the associated access token.
sub	The value of the sub parameter is the role and entity of the user, separated by a semicolon. For example, 1111;10.
aud	The value of the aud parameter is the integration record and the company, separated by a semicolon. Additionally, the client ID is a part of the aud parameter, separated by a comma. For example, 1A111AA1–AA11–1A11–1111–A1A1111111A1;1111, 661131f7bf0a2f8a4d09c79d3fa961eab66102dca43e07ad47d3a29628ced67b.
email_verified	As of NetSuite 2021.1, the value of the email_verified parameter is always `false`.
scope	The value of the scope parameter is either `openid`, `email`, or both, separated by a comma.
iss	The value of the iss parameter is `https://system.netsuite.com`.
exp	The value of the exp parameter represents the number of seconds until the token's expiration.
iat	The value of the iat parameter represents the number of seconds since the token was issued.
nonce	The value of the nonce parameter is the same as in Step One.
jti	The value of the jti parameter is the token ID, unique for every token.

Parameter Name	Description
email	The value of the email parameter is the user's email. If the `email` value is not a part of the scope parameter, the email parameter is not a part of the id token.

Note: The token's dot-separated values are Base64 encoded.

The signature is validated with a public key, which is associated with the kid parameter. To access public keys for your account, use the following URL:

`https://<accountID>.suitetalk.api.netsuite.com/services/rest/auth/oauth2/v1/keys`

where <accountID> represents your NetSuite account ID.

Certificate Rotation

As of NetSuite 2021.1, the certificates used to validate access and refresh tokens during the OAuth 2.0 code grant flow are no longer valid indefinitely.

The certificates are valid for 90 days and the system generates new certificates 30 days before the previous certificates expire.

The certificates are company-specific.

Troubleshooting NetSuite as OIDC Provider

Note: Applications authorized using the NetSuite as OIDC Provider feature in your NetSuite production account are not copied to your Release Preview account or to your sandbox accounts. Users must authorize applications explicitly in Release Preview and sandbox accounts to test the NetSuite as OIDC Provider feature in these accounts. Each time a sandbox account is refreshed, users again must authorize applications explicitly in the sandbox account.

See the following topics for NetSuite as OIDC Provider troubleshooting information:

- Authorization Code Grant Flow Errors
  - Authorization Errors in Step One
  - Response Errors in Step Two and in the Refresh Token Response
- NetSuite as OIDC Provider and the Login Audit Trail
  - Authorization Code Grant Flow Error Messages in the Login Audit Trail
  - Refresh Token Request Error Messages in the Login Audit Trail

For information about RESTlets and REST web services errors, see Troubleshooting OAuth 2.0.

Authorization Code Grant Flow Errors

For information about errors that may occur during the OAuth 2.0 flow for NetSuite as OIDC provider, see the following topics:

- Authorization Errors in Step One
- Response Errors in Step Two and in the Refresh Token Response

For information about RESTlets and REST web services errors, see RESTlets and REST Web Services Authentication Errors.

Authorization Errors in Step One

The following table lists errors that may occur in Step One of the OAuth 2.0 authorization code grant flow. Error requests are sent to the redirect URI with a specific error value, and should be handled by the application.

The redirect parameter is error.

Error Value	Error Description	Resolution
invalid_request	One or more required parameters are missing. **Important:** The redirect does not take place if the redirect URI in the GET request does not match the value in the **Redirect URI** field in the corresponding integration record. Only the error message should be displayed.	Ensure that none of the parameters is missing in the request in Step One. For more information, see Step One GET Request to the Authorization Endpoint.
unauthorized_client	The redirect does not take place if the client is unknown to the authorization server. Only the error message should be displayed.	Ensure that you use the correct values of the client ID and client secret for the corresponding integration record.
access_denied	A user clicks the **Deny** or **Back** button on the consent screen and interrupts the flow.	The user must Click **Allow** or **Continue** to give the application consent. For more information, see Consent Screen.
unsupported_response_type	The response type cannot be handled.	Ensure that the response type value is correct. For more information, see Step One GET Request to the Authorization Endpoint.
invalid_scope	The scope cannot be handled. The scope value is malformed, unknown, or invalid.	Ensure that the scope value is in correct format. For more information, see Step One GET Request to the Authorization Endpoint.

The following is an example of a redirect to the redirect URI with an error:

```
https://<your_redirect_uri>?
state=ykv2XLx1BpT5Q0F3MRPHb94j&role=1000&entity=12&company=1234567&error=<error_value>
```

For more information about Step One of the OAuth 2.0 authorization code grant flow, see Step One GET Request to the Authorization Endpoint.

Response Errors in Step Two and in the Refresh Token Response

The following table lists errors that may occur in Step Two of the OAuth 2.0 authorization code grant flow and in the response to the refresh token request.

The JSON format for the response is:

```
{
    "error": "<error_value>"
}
```

Error Value	Error Description	Resolution
invalid_request	Any of the following conditions may cause the invalid_request error to occur: ■ One or more required parameters are missing or malformed. ■ The grant_type value is incorrect. ■ Multiple client authentication approaches are used. ■ Any other type of a malformed request is sent. The HTTP status code is 400 Bad Request.	Ensure that your request is valid and in the correct format. For more information, see Step Two POST Request to the Token Endpoint.

invalid_client	Authentication of the client fails.
	The HTTP status code is 401 Unauthorized.
	The response header is set to:
	`Basic realm=<accountID>`
	Following is an example of the response header:
	`HTTP/1.1 401 Unauthorized`
	`WWW-Authenticate: Basic realm="123456"`
		Ensure that you use the correct values of the client ID and client secret for the corresponding integration record.

invalid_grant	Any of the following conditions may cause the invalid_grant error to occur: ■ The authorization code is invalid, expired, or revoked. ■ The refresh token is invalid, expired, or revoked. **Important:** In case the refresh token is expired, the application must go back to Step One of the OAuth 2.0 authorization code grant flow to restart the process. ■ The redirect URI does not match the redirect URI in the authorization request. ■ The authorization code or refresh token cannot be associated with the client. ■ The code_verifier parameter on Step Two does not match the code_verifier parameter in Step One. The HTTP status code is 400 Bad Request.	Ensure that values of all parameters are correct and matching the values from Step One of the flow. For more information, see Step Two POST Request to the Token Endpoint.
unauthorized_client	The value of the authorization grant_type is not allowed for the client.	Ensure that you use the correct values of the client ID and client secret for the corresponding integration record.

Error Value	Error Description	Resolution
unsupported_grant_type	The value of the grant_type parameter is neither `authorization_code` nor `refresh_token`. The HTTP status code is 400 Bad Request.	Ensure that the value of grat_type parameter is `authorization_code` or `refresh_token`.
invalid_scope	The scope cannot be handled. The scope value is malformed, unknown, or invalid. The HTTP status code is 400 Bad Request.	Ensure that the scope value is in correct format. For more information, see Step Two POST Request to the Token Endpoint.

For more information about Step Two of the OAuth 2.0 authorization code grant flow, see Step Two POST Request to the Token Endpoint.

For more information about the refresh token request, see Refresh Token POST Request to the Token Endpoint.

NetSuite as OIDC Provider and the Login Audit Trail

This section covers how to use the Login Audit Trail to track integrations and users, and provides details about error messages you may encounter.

For more information about tracking integrations and users, see Tracking NetSuite as OIDC Provider Integrations and Users.

For more information about error messages, see the following sections:

- Authorization Code Grant Flow Error Messages in the Login Audit Trail
- Refresh Token Request Error Messages in the Login Audit Trail

For information about RESTlets and REST web services, see RESTlets and REST Web Services Error Messages in the Login Audit Trail.

Tracking NetSuite as OIDC Provider Integrations and Users

You can use the Login Audit Trail to track NetSuite as OIDC Provider integrations and users.

To track integrations 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** and **Token-based Application Name**.
5. Click **Submit**.

   The **Detail** column displays error messages for any NetSuite as OIDC Provider logins with a status of Failure.

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

Authorization Code Grant Flow Error Messages in the Login Audit Trail

The following table lists errors that are visible in the **Detail** column of the Login Audit Trail.

Problem	Authorization Code Grant Flow Step One	Authorization Code Grant Flow Step Two	Resolution
The integration application does not use OAuth 2.0.	AuthorizationCodeGrant Required	AuthorizationCodeGrant Required	Ensure that the **Authorization Code Grant** box is checked in the corresponding integration record. For more information, see Create Integration Records for Applications that Use NetSuite as OIDC Provider for Outbound Single Sign-on.
Role or entity is inactive.	—	EntityOrRoleDisabled	Verify that the entity or role is active in the account.
The value of the state parameter is invalid.	InvalidState	—	Ensure that the value of the state parameter: ■ is 24 to 1024 characters long ■ consists of printable ASCII characters
Client ID or client secret is invalid.	UnknownIntegration	ClientAuthenticationFailed	Ensure that you use the correct values of the client ID and client secret for the corresponding integration record.
The value of the redirect URI parameter is invalid.	InvalidRedirectURI	—	Ensure that the redirect URI is a valid URL. for more information, see Create Integration Records for Applications that Use NetSuite as OIDC Provider for Outbound Single Sign-on.
The response type is invalid.	UnsupportedResponseType	—	The response type used is not valid for this step of the authorization code grant flow. For more information, see Step One GET Request to the Authorization Endpoint.
The user clicked Deny/Back on the consent screen.	AuthorizationExplicitly Denied	—	Start the OAuth 2.0 authorization code grant flow again and click Allow/Continue on the consent screen.
The value of the grant type parameter is either invalid or wrong.	—	InvalidGrantType	Ensure that the grant type value used is the correct one in the corresponding step of the authorization code grant flow. For more information, see OAuth 2.0 Authorization Code Grant Flow.
The NetSuite as OIDC Provider feature is	FeatureDisabled	FeatureDisabled	See Enable the NetSuite as OIDC Provider Feature.

Problem	Authorization Code Grant Flow Step One	Authorization Code Grant Flow Step Two	Resolution
not enabled in the account.			
The integration record is blocked.	IntegrationBlocked	IntegrationBlocked	Ensure that the value of the **State** field is set to **Enabled** on the corresponding integration record. For more information, see Create Integration Records for Applications that Use NetSuite as OIDC Provider for Outbound Single Sign-on.
Parameters for the Proof Key for Code Exchange (PKCE) are missing or malformed.	InvalidRequest	—	If you use PKCE in OAuth 2.0, make sure you configured the parameters correctly. For more information, see Step One GET Request to the Authorization Endpoint.
The code_verifier parameter in Step Two does not match the code_verifier parameter in Step One.	—	InvalidGrant	If you use PKCE in OAuth 2.0, make sure you configured the parameters correctly. For more information, see Step One GET Request to the Authorization Endpoint, and Step Two POST Request to the Token Endpoint.

Refresh Token Request Error Messages in the Login Audit Trail

The following table lists errors that are visible in the **Detail** column of the Login Audit Trail.

Problem	Refresh Token Request	Resolution
The access token is expired.	RefreshTokenExpired	Use the refresh token to get a new access token. If the refresh token is expired, initiate the authorization code grant flow to get a new pair of tokens. For more information, see OAuth 2.0 Authorization Code Grant Flow.
At least one of the following is invalid: ■ Entity ■ Contact ■ Role	EntityOrRoleDisabled	Verify that the entity, contact, or role exists in the account.
The signature is invalid.	InvalidSignature	Ensure that you use the correct certificate for token validation. For more information, see OAuth 2.0 Token Structure and Certificate Rotation.

Problem	Refresh Token Request	Resolution
		Warning: Invalidity of issuer or signature may be caused by cross-site request forgery (CSFR) attacks. To ensure that your application is safe, follow the OAuth 2.0 specification. For more information, see RFC6749 Section 10.12.
The integration application ID is invalid.	InvalidIntegration	Verify that the corresponding integration record exists in the account.
The integration application does not use OAuth 2.0.	AuthorizationCodeGrantRequired	Ensure that the **Authorization Code Grant** box is checked in the corresponding integration record. For more information, see Create Integration Records for Applications that Use NetSuite as OIDC Provider for Outbound Single Sign-on.
The scope value is empty in the token.	InvalidScope	Ensure that the structure of the access token is correct. For more information, see OAuth 2.0 Token Structure and Certificate Rotation.
Role or entity is inactive.	EntityOrRoleDisabled	Verify that the entity or role is active in the account.
Client ID or client secret is invalid.	ClientAuthenticationFailed	Ensure you use the correct values of the client ID and client secret for the corresponding integration record.
The value of the grant type parameter is invalid or wrong.	InvalidGrantType	Ensure that the grant type value used is the correct one in the corresponding step of the authorization code grant flow. For more information, see OAuth 2.0 Authorization Code Grant Flow.
The application attempted the refresh token request with an access token.	InvalidRefreshToken	Ensure that the application uses the access token for accessing RESTlets and REST web services, and the refresh token for the refresh token POST request. For more information, see Refresh Token POST Request to the Token Endpoint.
The NetSuite as OIDC Provider feature is not enabled in the account.	FeatureDisabled	See Enable the NetSuite as OIDC Provider Feature.
The integration record is blocked.	IntegrationBlocked	Ensure that the value of the **State** field is set to **Enabled** on the corresponding integration record. For more information, see Create Integration Records for Applications that Use NetSuite as OIDC Provider for Outbound Single Sign-on.

SAML Single Sign-on

SAML (Security Assertion Markup Language) is an XML-based standard that supports communication of user data among various 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. 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.

Note: To use SAML Single Sign-on, you need Full level for the SAML Single Sign-on permission. For more information, see the help topic NetSuite Permissions Overview.

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, 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 about 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 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 first steps in setting up SAML SSO are 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 > Setup Tasks > Enable Features and click the **SuiteCloud** subtab.
2. In the Manage Authentication section, check the **SAML Single Sign-on** box. Agree to the SuiteCloud Terms of Service when prompted.
3. Click **Save**.

> **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) 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 Access for Center Roles
- SAML SSO Permissions
- 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.

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. For more information about center roles access, see the following: SAML Single Sign-on Access to Web Store.

To add the SAML Single Sign-on permission to a customized center role:

1. Go to Setup > Users/Roles > User Management > Manage Roles.
2. Click **Edit** for a customized center role or click **Customize** 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 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. A user logging in with a role with 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.

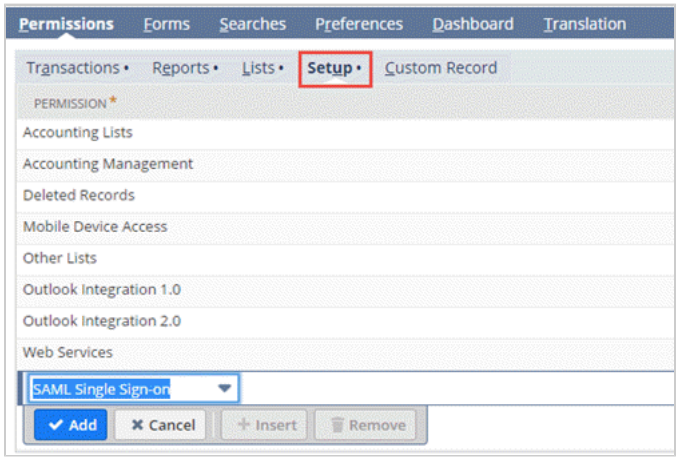
  For more information, see the help topic SAML Single Sign-on Access to Web Store

Both of these permissions are Setup type permissions that support only a Full 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 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 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 administrator has absolute responsibility for explicitly deciding who is allowed to access their NetSuite account using SAML Single Sign-on. The 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 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 Connect (OIDC) Single Sign-on permission.

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.

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.

To assign a SAML Single Sign-on role to users:

1. Find the appropriate entity record for the user. Go to Lists > Employees > Employees.
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

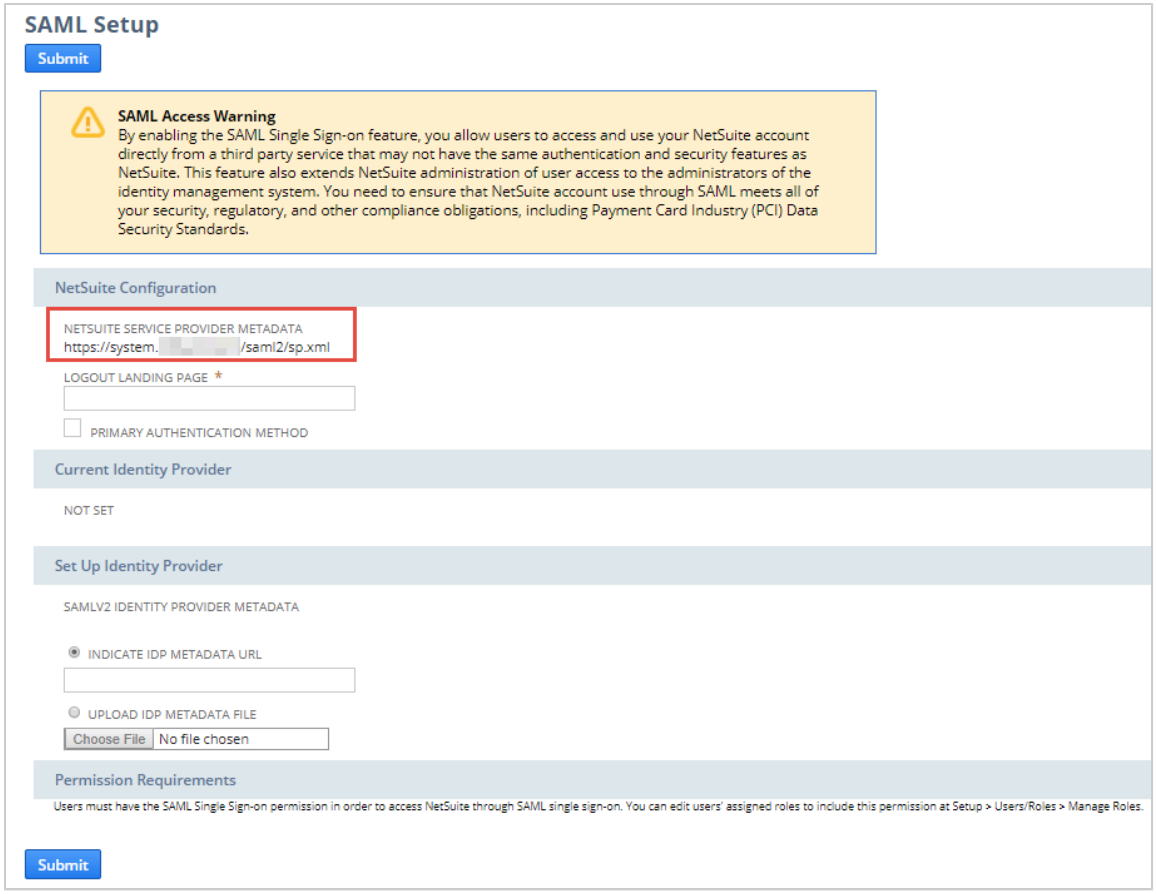
After the SAML Single Sign-on feature is enabled, administrators and users with the Set Up SAML Single Sign-on permission can view and edit the SAML Setup page in NetSuite. How you configure NetSuite as a service provider (SP) with the identity provider (IdP) of your choice depends on the IdP you have 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.

The person responsible for configuring SAML access to NetSuite on the IdP side should perform the following steps.

To copy the NetSuite SP metadata file and related URL:

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. Click the link in the **NetSuite Service Provider Metadata** field, and 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: As of May 2020, the default value for the location is set to the NetSuite system domain. You do not have to change the configuration if we move your account to a different data center location, or if you configure SAML SSO in multiple accounts in various data center locations.

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.

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
SP Entity ID	**Always refer to the NetSuite Service Provider Metadata file in your account.** Copy the SP entityID from the NetSuite Service Provider metadata file you downloaded from the SAML Setup page your account. The SP entityID is shown in the first line of the file.
Assertion Consumer Service	**Always refer to the NetSuite Service Provider Metadata file in your account.** Copy the URL from the NetSuite Service Provider metadata file you downloaded from the SAML Setup page in your account. **Important:** As of May 2020, the default Assertion Consumer Service refers to the NetSuite system domain: `https://system.netsuite.com/saml2/acs`. You do not have to change the configuration if we move your account to a different data center location, or if you configure SAML SSO in multiple accounts in various data center locations.
Single Logout Service	**Always refer to the NetSuite Service Provider Metadata file in your account.** Copy the URL from the NetSuite Service Provider metadata file you downloaded from the SAML Setup page in your account. **Important:** Use only the value on the first line of the list: `https://system.netsuite.com/saml2/slopost` Ensure you use a POST binding.

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.

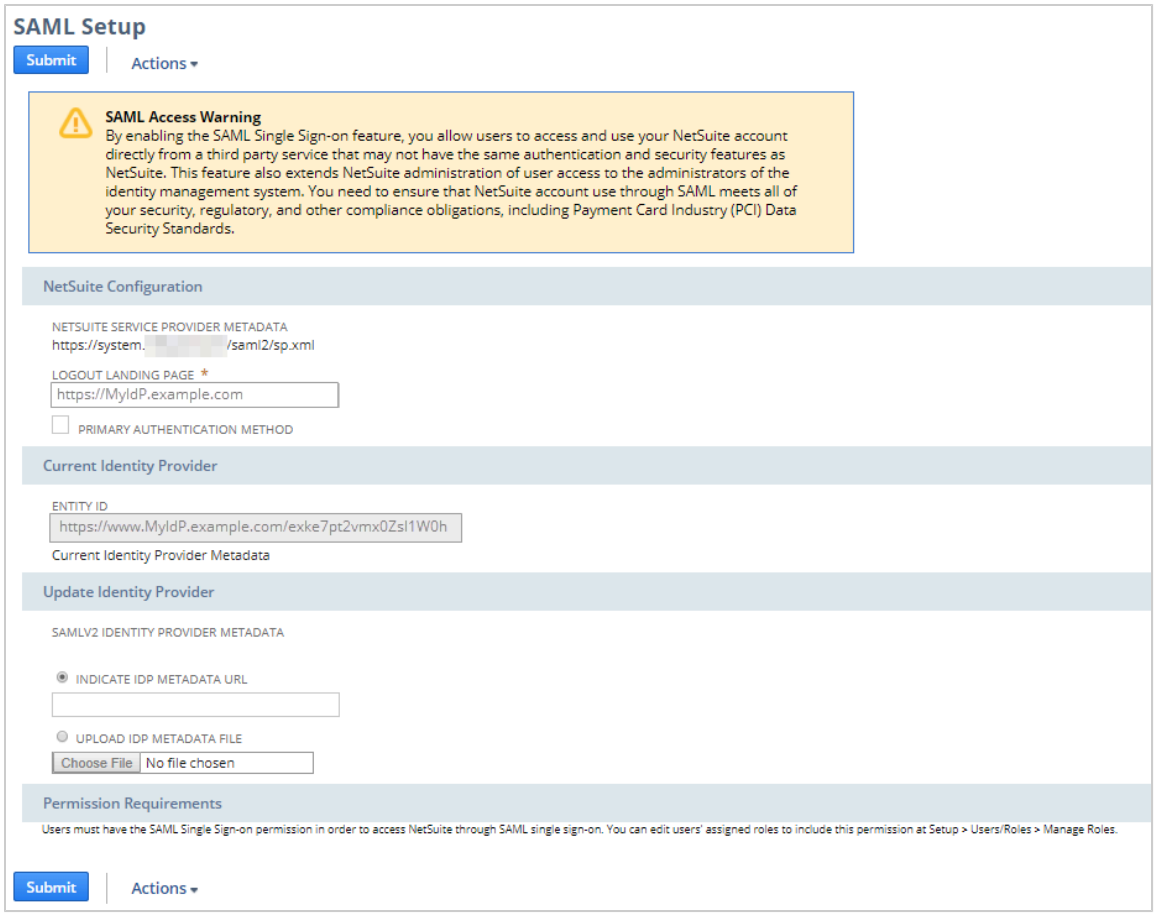
In many cases, the previous steps take care of all the information you need to provide to the IdP. For more information about signing assertions, encryption, and SAML attributes, see IdP Metadata and SAML Attributes.

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 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.)

Completing the SAML Setup Page

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: As of May 2020, the default value for the location is set to the NetSuite system domain. You do not have to change the configuration if we move your account to a different data center location, or if you configure SAML SSO in multiple accounts in various data center locations.

For details about completing the SAML Setup page, see:

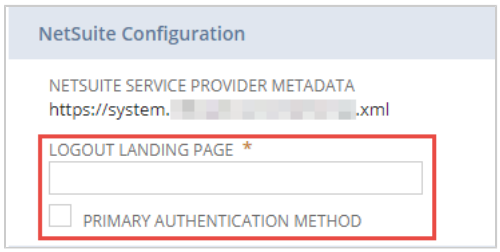
- Defining the NetSuite Configuration for SAML
- Set Up Your Identity Provider (IdP) in NetSuite

Note: To enable SAML access to a website (as opposed to the NetSuite application), you need to complete the SAML subtab of the Web Site Setup page. See the help topic SAML Single Sign-on Access to Web Store.

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** box 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 or as an account-specific domain, formatted like the following:

    https://system.netsuite.com/app/center/card.nl?c=<ACCOUNTID> or
    https://<accountID>.app.netsuite.com/app/center/card.nl

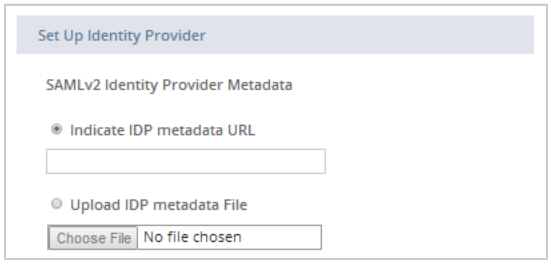
Note: If the **Primary Authentication box** is checked, and a user clicks a link containing the **c** or **compid** URL parameter or the account-specific domain URL, the user is redirected to the external IdP login page. The originally requested URL will be passed as a RelayState parameter, in accordance with the SAML 2.0 specification. This means that the IdP can direct the user back to the correct NetSuite resource after authentication. If there is a live session for the IdP, the user will be directed back to the NetSuite resource without being asked for credentials.

- Users will be redirected to the IdP login page upon session timeouts.

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 metadata 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**. Go 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, see Share SAML IdP Metadata in Multiple NetSuite Accounts.

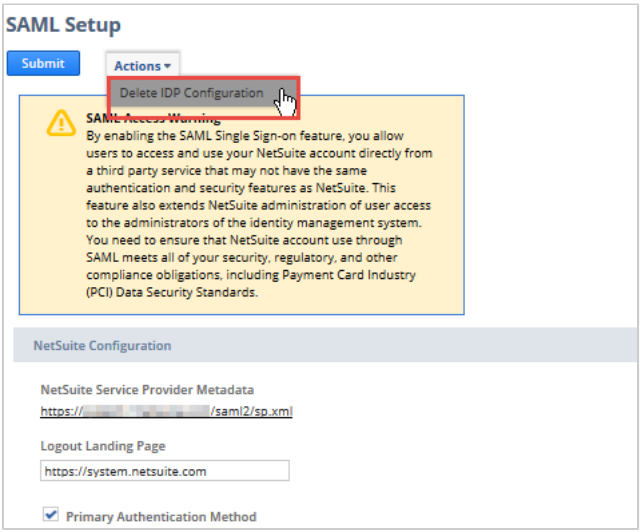
Remove the Current IdP Metadata

You can remove the current identity provider metadata without replacing it with another identity provider.

> **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**.



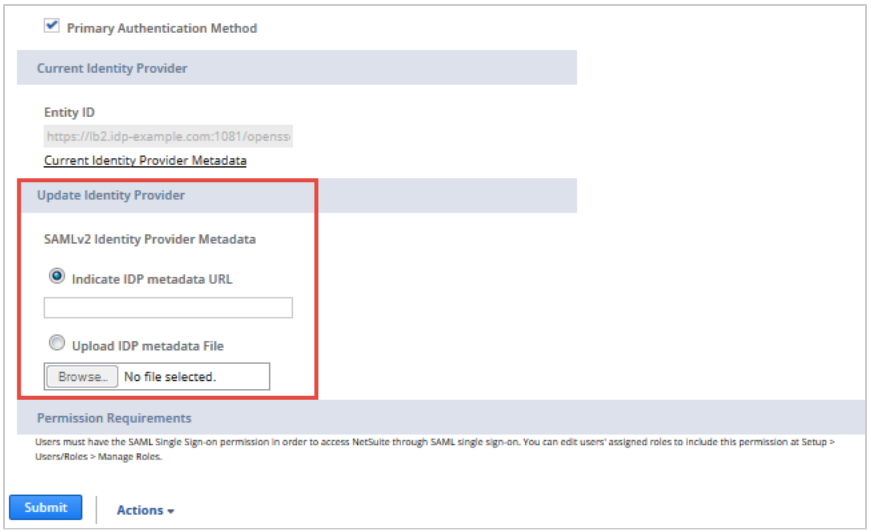
Note: For information about viewing or removing the identity provider metadata for SAML access to web stores, see the help topic SAML Single Sign-on Access to Web Store.

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**. Go to the location of the IdP configuration file you downloaded, select the file, and click **Open**.



Important: If your company uses SAML SSO in multiple accounts with a shared configuration, see Share SAML IdP Metadata in Multiple NetSuite Accounts.

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 various 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:

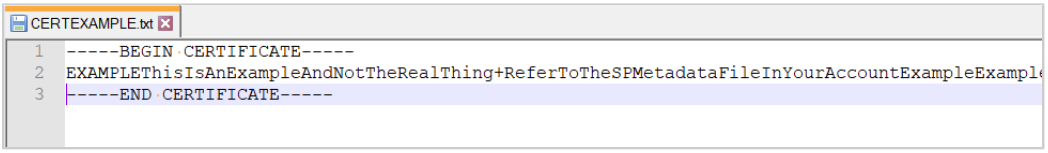
1. Download the SP metadata file from your NetSuite account.
    a. Go to Setup > Integration > SAML Single Sign-on.
    b. Download the SP metadata file to your computer. Remember the location you save the file to.
2. Create a new file in a text editor and enter the following text exactly as shown:

```
-----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.

SAML Attribute	NetSuite Parameter	Required or Optional
account	accountID	Optional, unless: ▪ you are sending the role attribute. ▪ you are sending the site attribute. ▪ access to both non-customer center and customer center SAML roles is needed. Sending the account attribute locks user access to a single account. See Account Attribute for more information.
role	role	Optional. See Role Attribute for more information.
site	site ID	Required for web store access. See Site Attribute for more information.
NameID or email	user email address	Required, must use the NameID attribute or the email attribute. See NameID and Email Attributes for more information.

SAML Attribute Statements

See the following sections for more information about SAML attributes.

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 for the account.

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.

Note: When the site attribute is provided, the user is directed to the web store with the 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.

The NetSuite system automatically generates and assigns IDs as the sites are created. If you do not know the site ID:

- For a Site Builder site, an administrator or a user with the Set Up Company permission can go to Commerce > Websites > Preview Website. The site ID is the **n** parameter at the end of the URL.

  The URL is in the following format:

  `http://shopping.<DataCenterID>.netsuite.com/s.nl?c=<accountID>&n=<siteID>`.

  For example, if your account was hosted in the US East data center, and your account ID was 123456, the URL for a Site Builder site would be:

  `http://shopping.na1.netsuite.com/s.nl?c=123456&n=1`.
- For a Suite Commerce Advanced site, an administrator or a user with the Set Up Company permission can go to Commerce > Websites > Website List. Click **Edit** for the required website. In the browser address bar, the site ID is the value of the **ID** parameter in the URL.

The URL is in the following format:

`https://<accountID>.app.netsuite.com/app/site/setup/siteadmin.nl?id=<siteID>&sitetype=ADVANCED&e=T`.

For example, if your account ID was 123456, and the site ID was 3, the URL for a Suite Commerce Advanced site would be:

`https://123456.app.netsuite.com/app/site/setup/siteadmin.nl?id=3&sitetype=ADVANCED&e=T`.

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.

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:

- 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: `<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:

```
...
<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:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">123456</saml:AttributeValue>
                </saml:Attribute>
```

```
  <saml:Attribute Name="role">
                    <saml:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">1010</saml:AttributeValue>
                </saml:Attribute>
                <saml:Attribute Name="site">
                    <saml:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">1</saml:AttributeValue><saml:AttributeStatement>
            </saml:AttributeStatement>
...
```

Interactions with NetSuite Using SAML

See the following topics for more information about SAML SSO interactions with NetSuite:

- SP-initiated and IdP-initiated Flows
  - The SP-initiated Flow
    - SAMLRequest and RelayState
- Single Logout (SLO)

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.

For more information about SAML SSO use with web stores, 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.

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 supports Single Logout (SLO) feature. The following Single Logouts (SLO) are supported:

- IdP-initiated SLO is supported for the NetSuite UI and Commerce web stores.

- SP-initiated SLO is supported for the NetSuite UI and Commerce web stores.

SAML SSO in Multiple NetSuite Account Types

If you are using SAML Single Sign-on (SSO) in more than one account type, see the following sections:

- Set Up and Configure SAML SSO in More Than One Account
- Enable SAML in Multiple NetSuite Account Types
  - Share SAML IdP Metadata in Multiple NetSuite Accounts

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 when using 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, and the refreshed sandbox has been activated.
- **Release Preview:** - You must configure SAML in your Release Preview account when it becomes available before each new NetSuite release.

Enable SAML in Multiple NetSuite Account Types

The following procedures do 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

Share SAML IdP Metadata in Multiple NetSuite Accounts

As of January 2020, there is a change to the way SAML IdP metadata is configured in multiple NetSuite accounts. You must complete a special procedure if you want to add a new account to the existing SAML configuration and share the same IdP metadata with this new account.

Important: Completing this procedure is only required if you want to add a new account that shares the same configuration with your current accounts. Multiple accounts can share the same IdP if the metadata files are identical.

For example, perhaps you currently use the same SAML metadata for your production and sandbox accounts. You decide you want to purchase another sandbox account and want to use the same SAML metadata in that new account. Or perhaps you want to set up SAML in your Release Preview account. You have two options for setting up SAML and sharing SAML metadata with additional NetSuite accounts.

- You should follow the procedure for the preferred option, see Redefine the IdP Configuration.
- However, if the preferred option is not feasible for your situation, see Upload an existing IdP metadata file to All New Accounts.

Important: If you do not follow one of the following procedures, you will encounter an error message if you only attempt to upload and save a metadata configuration file obtained from your IdP.

Redefine the IdP Configuration

This is the preferred approach for sharing the same SAML configuration in multiple NetSuite accounts.

To redefine the IdP configuration:

1. In a role with the Setup SAML Single Sign-on permission, or in an Administrator role, log in to a NetSuite account where the IdP metadata is shared.
2. Go to Setup > Integration > Manage Authentication > SAML Single Sign-on. Note the value in the read-only **Entity ID** field.
3. On the SAML Setup page under Actions, click **Delete IdP Configuration**. For more information, see Remove the Current IdP Metadata.

   Note: Make a list of all accounts from which you delete the IdP configuration file, meaning accounts that share the same **Entity ID** value.
4. Repeat steps 1-3 for each account that shares the same IdP configuration file.
5. Log in to the website of your IdP.
6. Locate the IdP metadata configuration file for the NetSuite application.
7. Copy the URL for this file or download the IdP metadata file from your IdP.

   Important: You must use this same file in the future when you add new accounts to your SAML configuration. If anything changes in the IdP metadata file, the IdP configuration must be redefined. Uploading an IdP metadata file containing any differences will generate a SAML Metadata Warning error message in the UI.
8. Refer to the list of accounts from which you deleted the IdP metadata. Log in to each account and go to Setup > Integration > Manage Authentication > SAML Single Sign-on. Either upload the IdP metadata file or point to the location (the URL) of the file from your IdP.

   Note: See Update the IdP Configuration File and Change Your IdP for NetSuite if you need more information about these options.
9. Log in to any new accounts you want to configure with the same IdP metadata and go to Setup > Integration > Manage Authentication > SAML Single Sign-on. Either upload the IdP metadata file or point to the location (the URL) of the file from your IdP

Upload an existing IdP metadata file to All New Accounts

If the preceding approach is not feasible in your situation, use the following option.

To upload an existing IdP metadata file (stored in NetSuite) to all new accounts:

1. In a role with the Setup SAML Single Sign-on permission, or in an Administrator role, log in to a NetSuite account where a SAML SSO is configured. (You should log in to your production account for this step.)
2. Go to Setup > Integration > Manage Authentication > SAML Single Sign-on.
3. Download the current IdP metadata file stored in NetSuite. In the Current Identity Provider section, open the Current Identity Provider Metadata file, go to the new tab, right-click and do a **Save As**.
4. In a role with the Setup SAML Single Sign-on permission, or in an Administrator role, log in to the new account you want to configure for SAML access.
5. Upload the IdP metadata file you downloaded in Step 3.

> **Important:** Repeat this procedure (starting with Step 4) for all of the new NetSuite accounts in which you want to share the same SAML configuration.

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 several 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 > Manage Authentication > 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. 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 about 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.

SAML SSO FAQ

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, and if 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 and others, 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. You should use the RSAwithSHA1 signature method.

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 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 Connect (OIDC) Single Sign-on

The OpenID Connect (OIDC) Single Sign-on feature provides several benefits for user access to the NetSuite UI and a web store. If the OIDC configuration is shared between various NetSuite accounts, users can switch between OpenID Connect (OIDC) Single Sign-on roles without requiring a separate login. User credentials and policies are managed by the OIDC provider (OP). NetSuite is the client, or relying party (RP).

OpenID Connect (OIDC) Single Sign-on is an alternative to SAML Single Sign-on. OIDC is an identity layer on top of the OAuth 2.0 protocol. OIDC uses JavaScript Object Notation (JSON) as the data format, and uses JSON Web Tokens (JWT) to transfer claims between parties.

Task List for OpenID Connect Single Sign-on Set Up

The following tasks must be completed to implement OpenID Connect (OIDC) Single Sign-on access to a NetSuite account. This is a list of basic tasks, more detail about each step is available in other topics in this section.

To implement OpenID Connect Single Sign-on to NetSuite:

1. Choose a vendor, an OpenID Connect provider (OP) and register NetSuite with your OP as the client, or relying party (RP). See Register NetSuite with Your OpenID Connect Provider.
2. Click the link in each of the following steps for information about how to complete the setup for the OpenID Connect (OIDC) Single Sign-on feature in NetSuite:
    a. Enable the OpenID Connect (OIDC) Single Sign-on Feature in NetSuite.
    b. Configure OpenID Connect (OIDC) in NetSuite.
    c. Customize Roles for OpenID Connect and add OpenID Connect Permissions.
    d. Assign the OpenID Connect Single Sign-on Role to Users.
    e. Tell your users how to access NetSuite using OpenID Connect. See User Access to NetSuite with OpenID Connect.

See also Troubleshoot OIDC for information about resolving OIDC-related errors.

If you are interested in setting up OpenID Connect (OIDC) access to Commerce web stores, familiarize yourself with the OIDC documentation in this section. Then, see the help topic OpenID Connect (OIDC) Access to Web Store.

Note: OIDC Identity Provider-initiated logout is not supported for both UI and Commerce. As of 2020.2, NetSuite supports Relying Party-initiated logout for UI and Commerce.

Important: if you are trying to implement **outbound** single sign-on, use the NetSuite as OIDC Provider feature. For more information, see NetSuite as OIDC Provider.

Register NetSuite with Your OpenID Connect Provider

To find a certified OpenID Connect provider (OP), go to https://openid.net/certification.

It is not possible to provide detailed instructions for configuring NetSuite as a client or relying party (RP) with your OIDC provider (OP). Refer to the documentation available from your OP for configuring OpenID

Connect access. However, see the following procedure for basic guidance on what must be accomplished to set up OpenID Connect access to NetSuite with your OP. The exact steps will vary, depending on the vendor you select as your OP.

Warning: Using OpenID Connect provider (OP) that is not certified might cause your OpenID Connect (OIDC) Single Sign-on configuration to work improperly.

To configure OpenID Connect with your OP:

1. Go to the website for your OP or use your on-premises administration console. Follow the instructions from your OP to register the NetSuite application as the relying party (RP).

   Note: Be aware of the following:

   - The only supported client authentication method is client_secret_basic.
   - The supported signing algorithms for ID tokens are RS256, RS384, and RS512.

2. It should be possible to specify more than one URI to redirect after login if a configuration is shared between multiple NetSuite accounts. The format for the login redirect URI is an account-specific domain URL in the following format:

   `https://<accountID>.app.netsuite.com/app/login/secure/oidclogin.nl`

   where <accountID> is a variable representing the NetSuite account ID.

3. Ensure that the email addresses of OP users are the same as the email addresses of the NetSuite users in your account. You must enter the email address of each NetSuite user who needs single sign-on capability to your OP. This ensures that your users are able to access NetSuite with OIDC.
4. When you have successfully completed registration with your OP, you are provided with a client ID and client secret, as well as a configuration URL. The format of the configuration URL is similar to this example:

   `https://<OPAcctID>.<OPdomain.com>/.well-known/openid-configuration`

   where <OPAcctID> and <OPdomain.com> represent variables.

   You will need the client ID, client secret and configuration URL values so that you can enter them on the OpenID Connect (OIDC) Single Sign-on setup page in NetSuite.
5. Assign an application or relying party to the OP users so that they will be able to access NetSuite.

Enable the OpenID Connect (OIDC) Single Sign-on Feature in NetSuite

A user with an Administrator role or a user that has the Enable Features permission can access the Enable Features page.

Warning: By enabling the OpenID Connect (OIDC) 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 OIDC meets all of your security, regulatory, and other compliance obligations, including Payment Card Industry (PCI) Data Security Standards.

To enable the OpenID Connect (OIDC) Single Sign-on feature:

1. Go to Setup > Company > Setup Tasks > Enable Features and click the **SuiteCloud** subtab.
2. In the **Manage Authentication** section, check the **OpenID Connect (OIDC) Single Sign-on** box. Click **I Agree** on the SuiteCloud Terms of Service when prompted.
3. Click **Save**.

Configure OpenID Connect (OIDC) in NetSuite

A user with an Administrator role or a user that has the Set Up OpenID Connect (OIDC) Single Sign-on permission can access the OpenID Connect (OIDC) Single Sign-on setup page. To complete the setup in NetSuite, you will need information from your OpenID provider (OP) when you registered NetSuite as the relying party (RP).

To configure OpenID Connect:

1. Go to Setup > Integration > Manage Authentication > OpenID Connect (OIDC) Single Sign-on.
2. Enter the **Client ID** you obtained from your OP.
3. Enter the **Client Secret** you obtained from your OP.
4. (Optional) Enter the **Post Logout Redirect URL** as a valid URL value.

   Important: The value of this field must match the value on OpenID Connect provider's (OP) side. A user is redirected to this URL after successful logout.

5. (Optional) Enter the **Allowed Email Domains** as comma-separated values.

   Important: If you leave this field blank, users with any email domain can access NetSuite using OIDC. If you want to restrict access to only specific email domains, list them in this field.

   - For instance, if your company's name was Example and your email domain was example.com, you would enter:

     `example.com`

   - However, some of your users may use different email domains to access your account. You should add those domains also. For instance:

     `example.com, gmail.com, <AnotherEmailDomain>.com`

6. The **Set Configuration From URL** option is selected by default. Enter the **Configuration URL** you obtained from your OP.

   Note: You should use the **Set Configuration From URL** option. However, if you choose the **Set Configuration Manually** option, you must gather the required information for the **Issuer**, **Authorization Endpoint**, **Token Endpoint**, and **Certificate URL** fields from your OP. Additionally, you can gather the required information for the **End Session Endpoint** field from your OP, if you want to use the relying party-initiated logout.

7. Click **Submit**.

You should see the following confirmation message in the UI: OpenID Connect (OIDC) configuration successfully saved.

If you receive an error message, see Troubleshoot OIDC for more information.

Important: The OIDC configuration is not shared between the NetSuite application and Commerce websites. An administrator must configure OIDC on the SSO tab of the website's setup page. Website users must be assigned the OpenID Connect (OIDC) Single Sign-on permission to log in to the website successfully. For more information, see the help topic OpenID Connect (OIDC) Access to Web Store.

Customize Roles for OpenID Connect

You might want to customize a standard NetSuite role (or roles) to use with OpenID Connect (OIDC) Single Sign-on permissions. You can also add these permissions to existing roles assigned to users that require this type of access.

To customize roles and add a permission:

1. Go to Setup > Users/Roles > User Managment > 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 OIDC.
4. Click the **Permissions** tab.
5. On the **Setup** subtab, select the appropriate OIDC permission from the list, and click **Add**. There are two OIDC permissions:
   - Set Up OpenID Connect (OIDC) Single Sign-on
   - OpenID Connect (OIDC) Single Sign-on

   See OpenID Connect Permissions for more information.
6. Click **Save**.

OpenID Connect Permissions

When the OpenID Connect (OIDC) Single Sign-on feature is enabled, the following permissions are available:

- **Set Up OpenID Connect (OIDC) Single Sign-on** - permits users other than those with an Administrator role (NetSuite administrators) to view and edit the OpenID Connect (OIDC) Single Sign-on setup page. The Administrator role already has this permission.

  Important: This is a highly-privileged permission, therefore two-factor authentication (2FA) is required. Roles with this permission are indicated in the Required 2FA column on the Two-Factor Authentication Roles page. For more information, see Two-Factor Authentication (2FA).
- **OpenID Connect (OIDC) Single Sign-on** - requires users to log in to the NetSuite UI using the OpenID Connect (OIDC) Single Sign-on feature. This permission must be explicitly assigned to a role.

Important: Be aware of the following:

- If a role is designated as Single Sign-on Only, 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.
- Users will receive notifications from NetSuite regarding password expiration. For more information, see the help topic Password Expiration Notifications.

See OpenID Connect Permission Limitations for more information.

Both OIDC permissions are Setup type permissions that support only a Full permission level.

To provide single sign-on access to users, the OpenID Connect (OIDC) 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.

Permissions are added to roles on the Role record page, available at Setup > Users/Roles > User Management > Manage Roles.

Important: After the OpenID Connect (OIDC) 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 the OpenID Connect (OIDC) Single Sign-on feature. This delay is related to caching; the new permission is not available until the cache has timed out.

For more information about adding permissions to roles, see the help topic Customizing or Creating NetSuite Roles.

OpenID Connect Permission Limitations

OpenID Connect (OIDC) roles and permissions have various limitations that are intended to prevent problems.

For example, the Administrator role does not have the OpenID Connect (OIDC) Single Sign-on permission and no user can log in with an Administrator role using the OpenID Connect (OIDC) Single Sign-on feature. This limitation ensures that an administrator can always log in and resolve any problems that might occur with the OIDC provider (OP) setup or with single sign-on access.

Another example of a limitation is that the OpenID Connect (OIDC) Single Sign-on permission cannot be added to a role that has SuiteAnalytics Connect permission. OpenID Connect (OIDC) Single Sign-on access is not supported for SuiteAnalytics Connect.

Some limitations are intended to ensure that the administrator has absolute responsibility for explicitly deciding who is allowed to access their NetSuite account using OpenID Connect (OIDC) Single Sign-on. The administrator is deciding to trust the OP to authenticate users and allow access to their NetSuite account.

- If a role is designated as Single Sign-on Only, a user with a role that has **OpenID Connect (OIDC) 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 the **OpenID Connect (OIDC) Single Sign-on** feature cannot access any roles that do not have **OpenID Connect Single Sign-on** permission.

- If a role has OpenID Connect (OIDC) Single Sign-on permission, it cannot also have SAML Single Sign-on permission.

Assign the OpenID Connect Single Sign-on Role 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.

Important: If a role is marked as Single Sign-on Only, a user with a role that has OpenID Connect (OIDC) Single Sign-on permission cannot log in directly to the NetSuite user interface on the standard NetSuite login page.

The following procedure is for adding a role with an OpenID Connect (OIDC) Single Sign-on permission to a user.

To assign an OpenID Connect (OIDC) Single Sign-on role to users:

1. Find the appropriate entity record for the user. For an employee, go to List > Employee > Employee.

   Note: If you need to create the user in NetSuite, see the help topic Manage Different Types of Users for links to information about setting up NetSuite access for various types of users (Employee, Vendor, and Partner).
2. Click the name of the user.
3. Click the **Access** subtab.
4. Click **Edit**.
5. On the Roles subtab, select your custom OpenID Connect role from the list.
6. Click **Add**.
7. Click **Save**.

User Access to NetSuite with OpenID Connect

Most users will access NetSuite with OpenID Connect one of two ways:

- On the portal of the OpenID Connect provider (OP), select NetSuite. Users are redirected to the NetSuite My Roles page.
- On the first login attempt, from an account-specific domain URL in one of the following formats:
  - `https://<accountID>.app.netsuite.com/app/login/secure/oidc.nl`, or
  - `https://<accountID>.app.netsuite.com/app/login/secure/oidcprivate.nl` for the Customer Center roles,

  where <accountID> is a variable representing the NetSuite account ID.

  When the OIDC configuration has been completed for an account, the user is redirected to the OP's login form. The user enters the OP login credentials, and is redirected to the My Roles page in NetSuite.

The user's roles are shown on the View My Roles page. Users should select the role with OpenID Connect (OIDC) Single Sign-on permission.

Note: After a user logged in successfully through OpenID Connect (OIDC) Single Sign-on, the user's preferred login method is remembered. If that user opens a deep link to a NetSuite resource when there is no active NetSuite session, the user is redirected to the OP login form.

Remove OpenID Connect Access to NetSuite

There are several ways to remove OpenID Connect access to NetSuite.

- Either of the following actions removes OpenID Connect access to NetSuite for a user or group of users in your account:
  - Removing the OpenID Connect (OIDC) Single Sign-on permission from the users' roles.
  - Editing the users' employee records in NetSuite to make a user or users inactive.
- The following action removes OpenID Connect access to NetSuite for all users in your account:
  - Disabling the OpenID Connect (OIDC) Single Sign-on feature in NetSuite.

Troubleshoot OIDC

This section provides details about OIDC error messages users might encounter. See the following topics for more information:

- OIDC Error Messages That May Be Encountered During Setup
- OIDC Error Messages Users May Encounter
- Resolving the Login Access Has Been Disabled Error

OIDC Error Messages That May Be Encountered During Setup

See the following table for information about error messages that you may encounter during the setup, and how to resolve them.

Error Message	Problem	Resolution
Unable to save your OpenID Connect (OIDC) configuration. Please review the configuration and correct any errors.	Causes of this error could be: - One of the fields on the OIDC Setup page contains a malformed URL.	To resolve this error, see Configure OpenID Connect (OIDC) in NetSuite. - Verify that all URLs are valid and entered correctly.
The URL <...> was unreachable. Ensure you have entered the correct URL, or use the **Set Configuration Manually** option.	Causes of this error could be: - The URL entered in the **Configuration URL** field is unreachable for some reason (the configuration URL is not accessible).	To resolve this error, see Configure OpenID Connect (OIDC) in NetSuite. - Identify the reason that the URL is unreachable, and resolve that problem.

Error Message	Problem	Resolution
	■ One of the fields on the form contains a malformed URL.	■ Verify that the Configuration URL is valid and entered correctly.

OIDC Error Messages Users May Encounter

See the following table for information about error messages that your may encounter when using OIDC, and how to resolve them.

There is a problem with the OpenID Connect (OIDC) configuration. Contact your administrator.	Causes of this error could be: ■ The OIDC feature is enabled, but the OIDC configuration in NetSuite has not been set up, the setup is incomplete, or is malformed.	To resolve this error: ■ A user with an Administrator role or a user that has the Set Up OpenID Connect (OIDC) Single Sign-on permission should verify that the setup is complete in NetSuite. See Configure OpenID Connect (OIDC) in NetSuite. ■ Validate your OIDC configuration with your OpenID Connect provider (OP). See Register NetSuite with Your OpenID Connect Provider.
The OpenID Connect (OIDC) Single Sign-on feature is not enabled in this account. Contact your administrator.	Causes of this error could be: ■ The OIDC feature is not enabled. ■ The OIDC feature is enabled, but your OIDC setup is not complete in NetSuite.	To resolve this error: ■ A user with an Administrator role or a user that has the Enable Features permission should verify that the feature is enabled. See Enable the OpenID Connect (OIDC) Single Sign-on Feature in NetSuite. ■ A user with an Administrator role or a user that has the Set Up OpenID Connect (OIDC) Single Sign-on permission should verify that the setup is complete. See Configure OpenID Connect (OIDC) in NetSuite.
The user <email address> does not exist. Contact your administrator to provision the user.	Causes of this error could be: ■ The user successfully authenticated at the OP, but the user does not exist in NetSuite.	To resolve this error, a user with an Administrator role may do the following: ■ If the user does not exist in NetSuite, create the user. See the help topic Manage Different Types of Users for links to information about setting up NetSuite access for various types of users (Employee, Vendor, Partner, and Customer). See also Assign the OpenID Connect Single Sign-on Role to Users.
The user <email address> does not have an assigned role. Contact your administrator.	Causes of this error could be: ■ The user successfully authenticated at the OP, and the user exists in NetSuite, but the user does not have a role assigned in NetSuite.	To resolve this error, a user with an Administrator role may do the following: ■ If the user exists but does not have an OIDC role, assign an OIDC role to the user. See Assign the OpenID Connect Single Sign-on Role to Users. See also Customize Roles for OpenID Connect.
The user <email address> does not have a role with OpenID Connect (OIDC)	Causes of this error could be:	To resolve this error, a user with an Administrator role may do the following:

permission. Contact your administrator.	▪ The user successfully authenticated at the OP, and the user exists in NetSuite, but the user does not have a role with the OpenID Connect (OIDC) Single Sign-on permission assigned in NetSuite.	▪ Assign an OIDC role to the user. See Assign the OpenID Connect Single Sign-on Role to Users. See also Customize Roles for OpenID Connect.
The user <email address> has an email domain name which is not permitted to access < account name> by OpenID Connect (OIDC) Single Sign-on. Contact your administrator.	Causes of this error could be: ▪ The user successfully authenticated at the OP, and the user exists in NetSuite, but the user's email domain name is not in the list of allowed domain names that can access your NetSuite account.	To resolve this error, a user with an Administrator role may do the following: ▪ If the user's email address is from a domain that should be able to access NetSuite, go to Setup > Integration > Manage Authentication > OpenID Connect (OIDC) Single Sign-on. Enter the user's email domain in the comma-separated list in the **Allowed Email Domains** field. See Configure OpenID Connect (OIDC) in NetSuite.
On the Insufficient Permissions page, users may encounter the following error message: The role <role name> <email address> you selected does not have OpenID Connect (OIDC) Single Sign-on permission. Contact your administrator.	Causes of this error could be: ▪ The user is attempting to access NetSuite with a role that does not have the OpenID Connect (OIDC) Single Sign-on permission.	To resolve this error: ▪ The user can switch to an appropriate OIDC role, if one is available on the Change Role list. If an appropriate role is not available, the user must contact the administrator. ▪ A user with an Administrator role can assign an OIDC role with the appropriate permission to the user. See Assign the OpenID Connect Single Sign-on Role to Users. See also Customize Roles for OpenID Connect.
On the Access Disabled page, users may encounter the following error message: Login access has been disabled for this role.	Causes of this error could be: ▪ The user is inactive. ▪ The role is inactive.	To resolve this error: ▪ Verify that the user is active. ▪ Verify that the role is active. See Resolving the Login Access Has Been Disabled Error.

Resolving the Login Access Has Been Disabled Error

Use the following procedures as needed to reactivate an inactive user or an inactive role.

Reactivate a User

See the following procedure for detailed information about how to reactivate a user.

To reactivate a user:

1. Open the appropriate record list page.
   - Lists > Employees > Employees
   - Lists > Relationships > Vendors
   - Lists > Relationships > Partners
   - Lists > Relationships > Customers

2. Click **Edit** beside the user record you want to reactivate.
3. Clear the **Inactive** box.
4. Click **Save**.

See the help topic Inactivating Users for information about how users are made inactive.

Reactivate a Role

See the following procedure for detailed information about how to reactivate a role.

To reactivate a role:

1. Go to Setup > Users/Roles > Manage Roles.
2. Check the **Show Inactives** box at the bottom of the list.
3. In the **Inactive** column, clear the box next to any role you want to reactivate.
4. Click **Submit**.

See the help topic Inactivating Roles for information about how roles are made inactive.

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 SuiteScript 2.x. 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.

You can manage digital signing using three SuiteScript 2.x 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. The following certificate file types are currently accepted:

- PFX
- P12
- PEM

Important: The certificate record holds information for a digital certificate, but it is not a standard NetSuite record and cannot be accessed with the N/record module.

Note: Regardless of user role, you cannot download digital certificates. Depending on which SuiteApps are installed in your account, you may see read-only system certificates in your list of digital certificates. These certificates are required for a secure connection to a third party service through a SuiteApp and cannot be edited or removed without uninstalling the SuiteApp.

To upload a new certificate:

1. Go to Setup > Company > Preferences > Certificates.
2. At the top of the page, click **Create New**.
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 a script ID for this certificate. The script ID of the certificate 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'.

   Important: Do not reuse a certificate ID if the certificate was deleted.
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 to save this certificate.
7. In the **Password** field, enter the password for this certificate. The 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. Click each name to select multiple employees. You do not need to use Ctrl or Command.

   Employees must also be using roles with the Certificate Access permission to be able to execute a script that accesses a certificate.
9. To restrict access through SuiteScript to specific scripts, enter the script IDs in the **Restrict to Scripts** field.

   For more information about restricting access to certificates, see Access to Digital Certificates.
10. In the **Subsidiaries** field, select which subsidiaries this certificate applies to. You can select more than one subsidiary, or you can check the box at the top of the list to select all subsidiaries. Selecting a subsidiary lets you search for certificates by subsidiary and does not affect access.
11. 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.
12. Check the **Copy Employees** box to copy additional employees on reminders. Select which employees to copy in the field below. Click each name to select multiple employees. You do not need to use Ctrl or Command.
13. 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 accounts, you must re-upload your certificate to the new account. 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 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 NetSuite 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. This permission is required for employees to execute a script, even if the employee is listed in the **Restrict to Employees** field on the certificate.

Locking and Restricting Certificates

To prevent the unauthorized use of a certificate, you can limit the use of the certificate to certain scripts and specific employees on the certificate record.

To restrict access to a certificate:

1. Go to Setup > Company > Preferences > Certificates.
2. Click the name of the certificate to open it in a new window.
3. Click the Audience tab.
4. To restrict access to the certificate to specific employees, check the **Restrict to Employees** box, and select employees who should be able access the certificate using SuiteScript. Employees must also have a role with the Certificate Access permission to use the certificate with SuiteScript. When you do not check the box, all employees with the Certificate Access permission can access the certificate with SuiteScript.
5. To restrict access to a specific script, enter the script's script ID in the **Restrict to Scripts** field. You can find the script ID for a script on the script record or in the **ID** column on the list of scripts at Customization > Scripting > Scripts. Separate multiple script IDs with commas.

   You can also set script restrictions with SuiteScript 2.x if you create the certificate object using the certificateControl.createCertificate(options) method.
6. Click Save.

In addition to these restrictions, you can use SuiteScript 2.x to lock and unlock Digital Certificates. Locking a certificate prevents anyone from accessing the certificate in the NetSuite UI, even the owner of the certificate.

You can use locking combined with script restrictions to restrict new scripts from using a certificate. For example, if you create a SuiteApp that contains a script and a certificate, you can restrict the certificate to only work with the SuiteApp's script and lock the certificate. You can then lock the script within the SuiteApp to block unauthorized access.

Use the following methods to lock and unlock a certificate:

- certificateControl.lock(options)
- certificateControl.unlock(options)

The following code sample uses a scheduled script to demonstrate how to lock a certificate. Because this sample uses `define` instead of `require`, you must deploy this script in your account. It is not meant to run in the SuiteScript Debugger.

```
/**
*@NApiVersion 2.x
*@NScriptType ScheduledScript
*/

define(['N/certificateControl'],function(certificateControl){
    function execute(context){
        //lock the certificate
        certificateControl.lock({
```

```
            //replace the following id with the script ID of your key
            id: 'custcertificate_testid'
        });
    }
    return{
        execute: execute
    }
});
```

SSH Keys for SFTP

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. For more information, see the help topic N/keyControl Module.

Note: Other than administrators, only users with custom roles that include View level for the Key access permission can upload or access keys information. For more information, see the help topic NetSuite Permissions Overview.

The following algorithms are supported:

- RSA
- DSA
- ECDSA

For more information, see Uploading Private SSH Keys and Access to SSH Keys.

Uploading Private SSH Keys

You can store and manage your keys on the Keys page. 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 PEM 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 when generating the key by using the `ssh-keygen` command.
8. In the **Audience** tab, check the **Restrict to Employees** box to restrict the usage of the key in SuiteScript to the specified list of employees.

   Employees must also be using roles with the Key Access permission to be able to execute a script that accesses a key.
9. To restrict access through SuiteScript to specific scripts, enter the script IDs in the **Restrict to Scripts** field.

   For more information about restricting access to keys, see Locking and Restricting Certificates.
10. Click **Save**. The key is decrypted and validated using the provided password. The key and password are securely stored to the NetSuite database.

Note: When testing in various accounts, you must re-upload your key to the new account. 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 SSH Keys

If you are not using the Administrator role, you need a custom role with the Key Management permission 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. Specific keys can be locked for viewing using the SuiteScript method keyControl.lock(options).
- **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. This permission is required for employees to execute a script, even if the employee is listed in the **Restrict to Employees** field on the key.

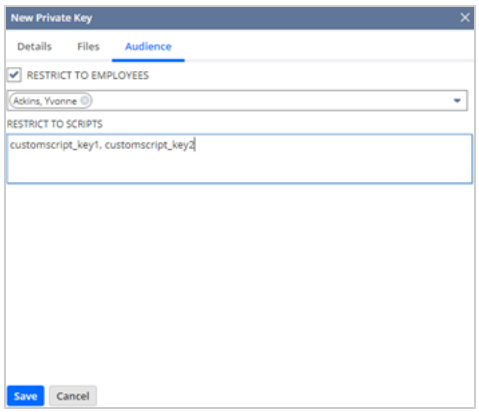
Locking and Restricting SSH Keys

To prevent the unauthorized use of a key, you can limit the use of the key to certain scripts and specific employees on the key record.

To restrict access to a key:

1. Go to Setup > Company > SSH Keys.
2. Click the name of the key to open it in a new window.
3. Click the Audience tab.
4. To restrict access to the key to specific employees, check the **Restrict to Employees** box, and select employees who should be able access the key using SuiteScript. Employees must also have a role with the Key Access permission to use the key with SuiteScript. When you do not check the box, all employees with the Key Access permission can access the key with SuiteScript.
5. To restrict access to a specific script, enter the script's script ID in the **Restrict to Scripts** field. You can find the script ID for a script on the script record or in the **ID** column on the list of scripts at Customization > Scripting > Scripts. Separate multiple script IDs with commas.

   You can also set script restrictions with SuiteScript 2.x if you create the key object using the keyControl.createKey(options) method..
6. Click Save.



When restrictions are set, you can use SuiteScript 2.x to lock and unlock SSH keys. Locking a key prevents anyone from accessing the key in the NetSuite UI, even the owner of the key. You can use locking combined with script restrictions to restrict new scripts from using a key. For example, if you create a SuiteApp that contains a script and a key, you can restrict the key to only work with the SuiteApp's script and lock the key. You can then lock the script within the SuiteApp to block unauthorized access.

Use the following methods to lock and unlock a key:

- keyControl.lock(options)
- keyControl.unlock(options)

The following code sample uses a scheduled script to demonstrate how to lock a key. Because this sample uses `define` instead of `require`, you must deploy this script in your account. It is not meant to run in the SuiteScript Debugger.

```
/**
*@NApiVersion 2.x
*@NScriptType ScheduledScript
*/

define(['N/keyControl'],function(keyControl){
    function execute(context){
        //lock the key
        keyControl.lock({
            //replace the following id with the script ID of your key
            id: 'custkey_testid'
        });
    }
    return{
        execute: execute
    }
});
```

Secrets Management

You can store, manage, and reference API secrets securely in NetSuite at Setup > Company > Preferences > API Secrets. You can then reference these secrets in third party integrations, preventing the need for plaintext secrets in scripts. API secrets include hashes, passwords, keys, and other secrets for managing digital authentication credentials. Secrets up to 1,000,000 characters are accepted.

Secret owners can set criteria to limit access for other users or allow access for a specific SuiteApp. Secrets are only referenced by script ID, and the password value cannot be displayed. A secret cannot be used by more than one SuiteApp. If you need to share a password across multiple SuiteApps, you must create multiple secrets with the same password.

Only SuiteScript 2.x APIs can use the secret. For a list of supported SuiteScript 2.x modules, see Supported SuiteScript 2.x modules.

Note: A secret can only be updated in the same account where it was created. Ensure that you do not enter a secret in an account that will be deleted in the future.

In SDF, you can add the customization object `custsecret_example_id.xml`, which contains the line `<secret scriptid="custsecret_example_id"/>` For more information, see the help topic SDF Custom Object and File Development in SuiteCloud Projects.

Other than administrators, only users with custom roles that include specific permissions for secrets access can manage secrets. For more information, see Access to Secrets.

Note: To maintain higher security in comparison to using the username and password combination, you can use an authentication API based on OAuth.

Supported SuiteScript 2.x modules

You can reference secrets by using any of the following modules:

Module	Method or Property
N/sftp	sftp.createConnection(options)
N/https	https.createSecretKey(options) https.createSecureString(options)
N/crypto	crypto.createSecretKey(options) SecretKey.secret
N/keyControl	keyControl.createKey(options) Key.password
N/certificateControl	certificateControl.createCertificate(options) Certificate.password

Creating Secrets

Warning: Do not use sensitive or private information in any of the informational fields in the UI. This information can be visible to other users.

You can store, manage, and reference API secrets securely in NetSuite at Setup > Company > Preferences > API Secrets. You can then reference these secrets in third party integrations, preventing the need for plaintext secrets in scripts. API secrets include hashes, passwords, keys, and other secrets for managing digital authentication credentials. Secrets up to 1,000,000 characters are accepted.

Tip: Take note of your old and new passwords. It can take up to one hour for the new password to be updated. When you have secrets for SFTP username and password, the old username and password must remain functional for at least one hour after secrets for the new username and password are entered.

To create a secret:

1. Go to Setup > Company > Preferences > API Secrets.
2. At the top of the page, click **Create New**.
3. In the **Create New Secret** window, on the **Details** tab, enter a descriptive name for this secret in the **Name** field. Do not use sensitive or private information. It is shown on the list of API Secrets.
4. In the **ID** field, enter a script ID for this secret. The ID of the secret lets you access it using SuiteScript. You should make this a descriptive ID with no spaces or special characters. NetSuite prefixes the script ID with 'custsecret'. Do not use sensitive or private information in this field. It is shown on the list of API Secrets.
5. Either type the secret into the **Password** field, or load it from a file. Multi-line secrets must be loaded from a file.
6. If you used the password field, enter your password again in the **Confirm Password** field.
7. (Optional) Check the **Expiration Warning** box if you want a warning to be displayed in the UI when the secret is nearing the expiration date.
8. In the **Description** field, enter a description of this secret. Do not use sensitive or private information. It is shown on the list of API Secrets.
9. (Optional) On the **Restrictions** tab, check the **Available To SuiteApp** box to reference this secret from a specified SuiteApp.
    a. In the **SuiteApp ID** field, enter the SuiteApp that is allowed to reference the secret. You can specify only one SuiteApp.
    b. In the **Allow On Test Accounts** field, enter account numbers that are allowed to reference the secret even if it is not included in a SuiteApp installed from the SuiteApp Marketplace. Separate multiple accounts with a comma.
10. (Optional) If you do not check the **Available To SuiteApp** box:
    a. In the **Restrict to Employees** field, select the users that are allowed to reference the secret using SuiteScript.
11. In the **Owners** field, select the users that are allowed to access and manage the secret.
12. Check the **Allow For All Scripts** box to allow any script in this account to access this secret using SuiteScript 2.x. If you clear this box, you must list script IDs for scripts that should have access in the **Restrict to Scripts** field.
13. In the **Restrict To Scripts** field, enter the script IDs that are allowed to reference the secret. Separate multiple script IDs with a comma.

14. Check the **Allow For All Domains** box to allow this decrypted secret to be sent to any domain. If you clear this box, you must list domains that should have access in the **Restrict to Domains** field.
15. In the **Restrict To Domains** field, enter the domains where decrypted passwords can be sent (applicable to SFTP and HTTPS only). Separate multiple domains with a comma. If you do not intend to use the secret with SFTP or HTTPS, consider adding an invalid domain to prevent the decrypted version of the secret from being sent or shared.
16. Click Save.

Filtering Secrets

The API Secrets page provides an overview of the secrets that are stored in NetSuite at Setup > Company > Preferences > API Secrets. You can enter keywords in the **Search** field to search for a secret by name, description, ID, application ID or owner.

There are also filters available to narrow down your choices when searching for a secret:

- **Exposure**
  - **All** - Displays all secrets.
  - **Local** - Displays secrets that are intended for local use only.
  - **Shared** - Displays secrets that are exposed to a SuiteApp.
- **Origin**
  - **All** - Displays all secrets.
  - **My** - Displays secrets that you own.
  - **This Account** - Displays secrets that are owned by any user in the account.
  - **External** - Displays secrets that were installed together with a SuiteApp.

Managing Secrets

You can edit details of secrets you own such as the name, password, and description. You can also edit the restrictions that are related to your secret. You can view the full audit trail of all operations done on a secret entry.

To manage a secret:

1. Go to Setup > Company > Preferences > API Secrets.
2. Select a secret.
3. In the **Secret** window, click one of the following tabs:
   - **Details** - Displays the details of the secret such as name, ID, and description.
   - **Restrictions** - Displays the restrictions of the secret.
   - **System Notes** - Displays the audit trail and life cycle of the secret. This tab is read-only.
4. Make your changes and click **Update**.

Access to Secrets

If you are not using the Administrator role, you need a custom role with the Secrets Management permission to view the API Secrets page and create a new secret.

The Secrets Management permission controls access to the API Secrets page in the UI.

Permission level	User with the permission level:
View	Can see secrets where the current user is listed as the owner
Create	■ Can create a new secret in the current account ■ Can make a secret he or she creates shareable with SuiteApps ■ Can set owners. The default owner is pre-set to the entry author ■ Can make a secret he or she owns shareable with SuiteApps
Edit	■ Can edit secrets where the current user is listed as the owner ■ Can make a secret he or she owns shareable with SuiteApps
Full	■ Can create secrets and set owners ■ Can see all secrets ■ Can set **Allowed for SuiteApps** ■ Can edit all secret entry fields, except for script ID ■ Can delete any secret created in this account ■ Can make any secret shareable with SuiteApps

Restricting Access to API Secrets

Secrets cannot be locked in the same way that SSH keys and certificates can be locked. However, you can restrict access to API secrets using the settings on the Restrictions tab of the secret in the UI.

You can restrict API secrets in the following ways:

- **Employees** — Employees you select in the **Owner** field can edit the secret and change the password. Employees must also either have Edit access to the Secrets Management permission with the role they are using or use the Administrator role. Roles with Full level of the Secrets Management permission can edit or delete any secret, even if they are not listed as an owner. Use the **Restrict to Employees** field to restrict secret decryption to specific employees. Only those employees listed can decrypt the secret when executing a script that uses the secret.
- **Scripts** — Clear the **Allow for All Scripts** box and enter script IDs in the **Restrict to Scripts** field to restrict a secret from bring decrypted using any scripts other than those listed.
- **SuiteApps** — To limit secret usage to a specific SuiteApp, check the **Available to SuiteApp** box, and enter the SuiteApp ID where the secret can be accessed. If the secret should be accessible from specific accounts for testing purposes, enter the account numbers if the **Allow On Test Accounts** field.
- **Domains** — Clear the **Allow for All Domains** box and enter domains in the **Restrict to Domains** field to restrict a secret from being decrypted on any domain other than those listed. If you do not want the secret decrypted on any domain, you can enter an invalid domain name.

Code Sample

The following code sample shows how to reference a secret by using its script ID.

```
require(['N/file', 'N/sftp', 'N/certificateControl'], (file, sftp, certificateControl) => {
    var connection = sftp.createConnection({
```

```
        username: 'sftpuser',
        secret: 'custsecret1', // This is the reference to the script ID.
        url: '172.25.182.109',
        port: 22,
        directory: 'certificates'
    });

    try {
        var cert = certificateControl.createCertificate({
            name: 'NAAcert2020',
            file: connection.download({
                directory: '2020',
                filename: 'NAA_cert.pfx'
            }),
            password: 'custsecret2'   // This is the reference to the script ID.
        });
        var certId = cert.save();
    } catch (e) {
        log.error(e.message);
    }
});
```

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
- Setting up OAuth 2.0 for a RESTlet Integration
- Using User Credentials for RESTlet Authentication

Note: OAuth 2.0 is the preferred authentication method. You should consider using OAuth 2.0 instead of TBA whenever possible.

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** box checked, 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. For information about account-specific domains for RESTlets, see the help topic Integration Domains. To handle this task, you can also 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) or OAuth 2.0 to access resources on behalf of a user, eliminating the need to share login credentials such as username and password. You should use TBA or OAuth 2.0 for RESTlet authentication. For more information, see the following topics:
  - The Three-Step TBA Authorization Flow for TBA.
  - OAuth 2.0 for Integration Application Developers for OAuth 2.0.

  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. 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.

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.

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.

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, or two-factor authentication. To enable token-based authentication, see Enable the Token-based Authentication Feature.

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.

For code samples and examples of signature creation and token-based authentication, see OAuth Library Consumption for Client Application and Create oauth_signature for Token-based Authentication in RESTlet.

For information about calling a token endpoint to issue or revoke a token, see Issue Token and Revoke Token REST Services for Token-based Authentication in the Token-based Authentication section of the Help Center.

Using TBA for RESTlet Authentication (OAuth)

If appropriate, you can use the 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
- The Three-Step TBA Authorization Flow
- Example OAuth Header
- Tracking RESTlet Calls Made with TBA and OAuth 2.0

Note: If you are calling a RESTlet from an external source, you must authenticate by using TBA, OAuth 2.0 or user credentials. For details on user credentials, see Using User Credentials for RESTlet Authentication. For details on OAuth 2.0, see Setting up OAuth 2.0 for a RESTlet Integration.

TBA Setup Requirements

Using the OAuth protocol with RESTlets requires the 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 Enable the Token-based Authentication Feature.

- You must have created a role that permits logging in using token-based authentication. For details, see Set Up Token-based Authentication Roles.
- You must have assigned a user to a role that has permission to log in by using token-based authentication. For details, see Assign Users to Token-based Authentication Roles.
- An integration record representing the sending application must exist at Setup > Integration > Manage Integrations. On the integration record, the **Token-based Authentication** box must be checked. Enabling this option causes the system to generate the consumer key and secret that represent the application. For details, see Create Integration Records for Applications to Use TBA.
- You must have the consumer key and secret that were generated when the integration record's **Token-based Authentication** box was checked. If you do not have these credentials, you can generate new ones. For details, see the help topic 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 Manage 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 Step One Obtain An Unauthorized Request Token.

Example OAuth Header

Warning: The end of support for the HMAC-SHA1 signature method is targeted for 2023.1. You should update your integrations to use the HMAC-SHA256 signature method as soon as possible.

The following code sample 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="4a3ff6c251a55057bb1e62d8dc8998a0366e88f3a8fe735265fc425368b0f154",
   oauth_token="52cfe88fecf2e2b74e833e7dfc4cae79ff44c3ca9f696d61e2a7eac6c8357c3c",
   oauth_nonce="qUwlmPvtGCS4sHJe8F7x",
   oauth_timestamp="1462453273",
   oauth_signature_method="HMAC-SHA256", oauth_version="1.0",
   oauth_signature="8PI9lIYxUmUONjxEFJUSMD9oOmc%3D"
```

Setting up OAuth 2.0 for a RESTlet Integration

Note: OAuth 2.0 is the preferred authentication method. You should consider using OAuth 2.0 instead of TBA whenever possible.

NetSuite supports OAuth 2.0, a robust authorization framework. This authorization framework enables client applications to use a token to access NetSuite through REST web services and RESTlets. The application accesses the protected resources on behalf of a user who gave an explicit permission for the access. This method eliminates the need for RESTlets or REST web services integrations to store

user credentials. OAuth 2.0 can be used as an alternative to token-based authentication. It is more straightforward to implement, because request signing is not required.

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 OAuth 2.0.

OAuth 2.0 allows integrations to comply with any authentication method that is deployed in a NetSuite account for UI login, such as SAML Single Sign-on, OpenID Connect (OIDC) Single Sign-on, or Ttwo-factor authentication. To enable OAuth 2.0 feature, see Enable the OAuth 2.0 Feature.

OAuth 2.0 introduces two new permissions. For more information, see Set Up OAuth 2.0 Roles.

Administrators and users with the OAuth 2.0 Authorized Applications Management permission can manage all authorized applications in the account. For more information, see Managing OAuth 2.0 Authorized Applications.

You can choose what flow to set up for the OAuth 2.0 feature in your account. For more information, see OAuth 2.0 for Integration Application Developers.

Using OAuth 2.0 for RESTlet Authentication

You can use the OAuth 2.0 feature to authenticate RESTlets' access to NetSuite. With this approach, you use the OAuth 2.0 authorization framework to construct an authorization header. For details, see the following topics:

- OAuth 2.0 Setup Requirements
- OAuth 2.0 Authorization Code Grant Flow
- OAuth 2.0 Authorization Header

OAuth 2.0 Setup Requirements

Before you can use the OAuth 2.0 authorization framework, you must complete the following tasks:

- Enable the OAuth 2.0 feature in your account. For more information, see Enable the OAuth 2.0 Feature.
- Set up roles for use with OAuth 2.0 and assign users to the roles. For more information, see Set Up OAuth 2.0 Roles.
- Create an integration record for use with OAuth 2.0. For more information, see Create Integration Records for Applications to Use OAuth 2.0.

After you set up an integration record for use with OAuth 2.0, you must create an external application that initiates the OAuth 2.0 flow. For more information, see OAuth 2.0 for Integration Application Developers.

OAuth 2.0 Authorization Header

After you finish the authorization code grant flow and the application is granted an access token, see the following information to create the OAuth 2.0 authorization header.

The format of the URL is:

```
https://<accountID>.app.netsuite.com/app/site/hosting/restlet.nl?script=1&deploy=1
```

The structure of the authorization header is:

```
Authorization: Bearer <access token>
```

The following is an example of the OAuth 2.0 authorization header for RESTlets:

```
Authorization: Bearer eyJraWQiOiIyMDIwXzEiLCJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdWIiOiIxMDAwOzEyIiwiYXVkIjoiN0VCODkwREMtNEJDRC00RTQ5LTkzNDEtRjZEMDIyNDUxOEY5OzM4Mjk4NTUiLCJ0dHlwZSI6IkFDQ0VTUyIsInNjb3BlIjpbIlJFU1RMRVRTIl0sImlzcyI6Imh0dHBzOlwvXC9zeXN0ZW0ubmV0c3VpdGUuY29tIiwiZXhwIjoxNTgwODI1NjQyLCJpYXQiOjE1ODA4MjIwNDJ9.sTNSUlE1w-X_zhNPou_pRvHPob_p6iTkvA329yfVqrFFcgy0Ma14HA1WtlYmd8Xy8TGvC5str_ZYEBNq9adNSb1inkgB4orFCus5plvCzuLaeA_kYWc6KEFq6Z2jfBBymrDtLqujvvBMxNan88KN0UXM7CaNDGrg7tUllcQcB6mJwiqrRMXPWPXSZMc17CgroIPwvNCaF7mK9np4V-s0nhlCCII_XuESWXZom2nJtserwiLC7db2psrmtXKSu0l75XRYWb8Qn1G3x56oYz56TAfjB2bM6kUYq-s4Io2QHHdD0HxZSH-d_i5gY3sfCIqzr9Z4G8u6IHLN0fThDTt3hQ
```

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.

Warning: As of the **2021.1 release**, user credentials authentication for newly created RESTlets are not supported. If you attempt to authenticate a new RESTlet, with user credentials after your account is upgraded to 2021.1, an HTTP error response is returned

Either the three-step TBA authorization flow or the OAuth 2.0 should be used for all new integrations. Developers of existing integrations currently using the issuetoken endpoint should consider migrating the integration to the TBA authorization flow. For information, see the following:

- The Three-Step TBA Authorization Flow for TBA.
- OAuth 2.0 for Integration Application Developers for OAuth 2.0.

If you are calling a RESTlet from an external source, you must authenticate by using token-based authentication or OAuth 2.0. For details on TBA, see Using TBA for RESTlet Authentication (OAuth). For details on OAuth 2.0, see Using OAuth 2.0 for RESTlet Authentication.

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.

Field	Description	Required?	Notes
nlauth_account	The ID of the NetSuite account where the RESTlet is deployed.	Yes	—
nlauth_email	The email address with which the user logs in to NetSuite.	Yes	—

Field	Description	Required?	Notes
nlauth_signature	The user's password.	Yes	—
nlauth_role	The internal ID of a role with which the user is associated.	No	If you omit this value, the system selects a role based on the logic described in RESTlet and SOAP Web Services Role Selection Logic.
nlauth_application_id	The application ID of the integration associated with the RESTlet.	No	The application ID is optional, but you should use it. By adding an application ID, you associate RESTlet requests 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.
nlauth_otp **Important:** This parameter can only be used with the issue token endpoint.	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: ■ Required 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

As of 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 for the account, 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 for the account, if the default role is a Customer Center role.
6. The last Customer Center role that the user logged in with.

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 about 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 sample 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 sample 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_role=37, nlauth_otp=654321
```

For an example of a shell script that generates an NLAuth header, see the help topic Example of Shell Script that Calls a RESTlet.

Tracking RESTlet Calls Made with TBA and OAuth 2.0

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 about using integration records in conjunction with RESTlets, see the following topics:

- Create Integration Records for Applications to Use TBA
- Blocking an Application
- 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.

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. Go to Setup > Integration > Managing Integrations, and open the appropriate integration record for editing.
2. Set the **State** field to **Blocked**.
3. Click **Save**.

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.

Issue Token and Revoke Token REST Services for Token-based Authentication

You can call a token endpoint to issue a token, to revoke a token, and to obtain information about a token. See the following sections:

- Calling a token endpoint to issue a token
- Calling a token endpoint to revoke a token
- Calling a token endpoint to obtain user information based on a token

Important: You can use TBA with those integrations that require the Administrator role. Administrators can only create tokens for their own use by clicking the Manage Access Tokens link in the Settings portlet, or by using the token endpoint.

In addition to creating a token manually through the NetSuite UI, developers and users can issue or revoke their own tokens programmatically using a token endpoint. You can also use a token endpoint to obtain information about a token.

Use the appropriate domain to call the token endpoint:

- RESTlet domain: `https://<accountID>.restlets.api.netsuite.com/`
- system domain: `https://<accountID>.app.netsuite.com`

Users cannot programmatically issue or revoke tokens for other users using a token endpoint. For information about creating tokens for other users through the NetSuite UI, see Viewing, Editing, Creating, and Revoking TBA Tokens.

Account-specific domains are supported for RESTlets. For example, if your account ID is 123456, your account-specific REST domain would be `123456.restlets.api.netsuite.com`. For more information, see the help topic URLs for Account-Specific Domains. See also the Integration Domains section in the topic How to Transition from Data Center-Specific Domains.

Important: Whether using The Three-Step TBA Authorization Flow, or 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 **Require Approval during Auto-Installation of Integration** box 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.

Calling a token endpoint to issue a token

- Use the NetSuite NLAuth authorization header. The token is created under the role specified in the NLAuth authorization header. For more information, see Using User Credentials for RESTlet Authentication.
- Parameters must be Url encoded. This is particularly important for parameters which include special characters like spaces, for example, token name.

- A token endpoint consumes two GET parameters. For an issuetoken request, the Consumer Key parameter is required, and the Name (the name of the token) is optional.

  For example:

```
https://<accountID>.restlets.api.netsuite.com/rest/issuetoken?consumerKey=<CONSUMER_KEY>&name=<TOKEN_NAME>
```

- The issue token endpoint has been extended to accommodate the 2FA requirement for highly privileged roles. There is an optional parameter, nlauth_otp, that you can include in the NLAuth Authorization header. For more information, see Required 2FA, the IssueToken Endpoint, and nlauth_otp.

Calling a token endpoint to revoke a token

- In a call to a token endpoint to revoke a token, use either:
  - The NetSuite NLAuth authorization header. See Using User Credentials for RESTlet Authentication in the topic Authentication for RESTlets.
  - The OAuth authorization header. See the topic Authentication for RESTlets.
- A token endpoint consumes two GET parameters. For a revoketoken request, both the consumerKey parameter and the token parameter are required.

  Here is an example of a request: `https://<accountID>.restlets.api.netsuite.com/rest/revoketoken?consumerKey=<CONSUMER_KEY>&token=<TOKEN>`

Calling a token endpoint to obtain user information based on a token

The tokeninfo endpoint returns information about a user based on the access token. The endpoint is `https://<accountID>.restlets.api.netsuite.com/rest/tokeninfo`, where <accountID> is a variable for the company's account ID. A response to a GET request contains data in JSON format, including information such as:

- Company Name
- Company ID (account ID)
- Role Name
- Role ID
- Entity ID