

# Security

## Table of Contents

Table of Contents .....	1
Topics in this section:.....	4
Realms .....	4
Privileges .....	6
Privilege Types .....	7
Privilege Actions .....	9
Example: Creating an Application Privilege .....	10
Example: Creating a Repository View Privilege .....	10
Roles .....	11
Mapping External Groups to Nexus Roles .....	14
Users .....	14
Anonymous Access .....	17
LDAP .....	18
Enabling the LDAP Authentication Realm .....	18
LDAP Connection and Authentication .....	19
User and Group Mapping.....	21
Security Setup with User Tokens .....	26
Enabling and Resetting User Tokens.....	26
Accessing User Tokens in Realms .....	27
Accessing and Using Your User Token .....	27
Authentication via Remote User Token .....	28

Configuring SSL .....	29
Outbound SSL - Trusting SSL Certificates of Remote Repositories .....	29
Outbound SSL - Trusting SSL Certificates Globally .....	31
Outbound SSL - Trusting SSL Certificates Using Keytool.....	33
Inbound SSL - Configuring to Serve Content via HTTPS .....	34
Using A Reverse Proxy Server .....	34
Serving SSL Directly .....	35
 Auditing .....	 37

Nexus Repository Manager Pro and Nexus Repository Manager OSS use role-based access control that gives administrators very fine-grained control over user rights to:

- read from a repository or a subset of repositories
- administer the repository manager or specific parts of the configuration
- access specific parts of the user interface
- deploy to repositories or even just specific sections of a repository

The default configuration ships with roles and users with a standard set of permissions. As your security requirements evolve, you will likely need to customize security settings to create protected repositories for multiple departments or development groups. Nexus Repository Manager provides a security model that can adapt to any scenario.

- ✓ The default administrator user give you full control and uses the username **admin** and the password **admin123**.

This section covers all aspects of security of the repository manager including:

- user account and access right management related to user interface as well as to component access documented in [Privileges](#)(see page 6), [Roles](#)(see page 11) and [Users](#)(see page 14)
- selection of security backend systems called [Realms](#)(see page 4) including the built-in system as well as LDAP and others
- management of SSL certificates from remote repositories, SMTP and LDAP servers documented in [Configuring SSL](#)(see page 29)

Security-related configuration can be performed with the feature views available via the *Security* section of the *Administration* main menu. Many of the features shown in this section are only available to users with the necessary privileges to access them.

The role-based access control system is backed by different authentication and authorizations systems as documented in [Realms](#)(see page 4) and designed around the following security concepts:

### Privileges

Privileges are rights to read, update, create, or manage resources and perform operations related to the user interface as well as the components managed by the repository manager in the various repositories. The repository manager ships with a set of core privileges that cannot be modified.

### Roles

Privileges can be grouped into collections called roles to make it easier to define privileges common to certain classes of users. For example, administrative users will all have similar sets of permissions. Instead of assigning individual privileges to individual users, you use roles to make it easier to manage users with similar sets of privileges.

### Users

Users can be assigned one or more roles, and model the individuals who will be logging into the user interface and read, deploy, or manage repositories as well as connect from client tools such as Apache Maven.

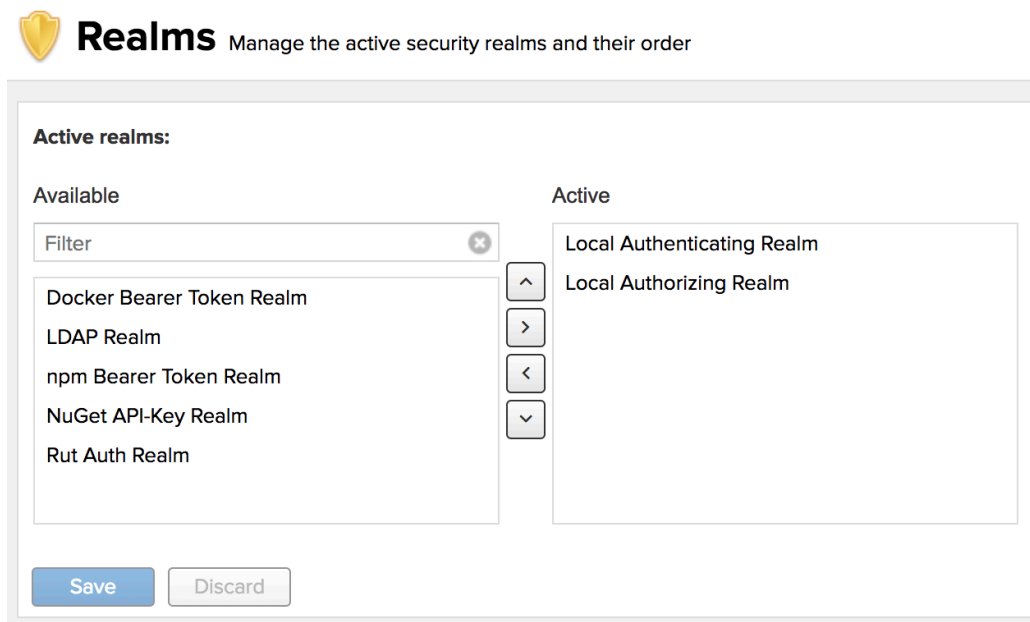
## Topics in this section:

- [Realms](#)(see page 4)
- [Privileges](#)(see page 6)
- [Roles](#)(see page 11)
- [Users](#)(see page 14)
- [Anonymous Access](#)(see page 17)
- [LDAP](#)(see page 18)
- [Security Setup with User Tokens](#)(see page 26)
- [Authentication via Remote User Token](#)(see page 28)
- [Configuring SSL](#)(see page 29)
- [Auditing](#)(see page 37)

## Realms

**Available in Nexus Repository OSS and Nexus Repository Pro**

The feature view for security realms administration displayed in *Figure 6.1, "Security Realms Administration"* allows you to activate and prioritize security realms used for authentication and authorization by adding them to the *Active* list on the right and placing them higher or lower on the list. It can be accessed via the *Realms* menu item located under *Security*, in the *Administration* main menu.



**Figure 6.1. Security Realms Administration**

Effectively, this configuration determines what authentication realm is used to grant a user access and the order the realms are used.

### Local Authenticating Realm and Local Authorizing Realm

These are the built-in realms used by default. They allow the repository manager to manage security setup without additional external systems.

### Crowd Realm

This realm identifies external storage in an Atlassian Crowd system with details documented in [Atlassian Crowd Support](https://help.sonatype.com/display/NXRM3/Atlassian+Crowd+Support)<sup>1</sup>.

### Docker Bearer Token Realm

This realm permits docker repositories with the ability to have anonymous read enabled on their repositories in conjunction with the *Force basic authentication* configuration setting. This is documented further in [the Docker section](#)<sup>2</sup>.

### LDAP Realm

This realm identifies external storage in an LDAP system including e.g., Microsoft ActiveDirectory, ApacheDS, OpenLDAP with details documented in [LDAP](#)([see page 18](#)).

### npm Bearer Token Realm

<sup>1</sup> <https://help.sonatype.com/display/NXRM3/Atlassian+Crowd+Support>

<sup>2</sup> <https://help.sonatype.com/display/NXRM3/Private+Registry+for+Docker#PrivateRegistryforDocker-AnonymousReadAccess>

This realm permits users with previously generated bearer tokens to publish *npm* packages. See [npm Security](#)<sup>3</sup> to learn how to establish a connection in order to publish.

#### NuGet API-Key Realm

This realm is required for deployments to NuGet repositories as documented in [.NET Package Repositories with NuGet](#)<sup>4</sup>.

#### Rut Auth Realm

This realm uses an external authentication in any system with the user authorization passed to the repository manager in a HTTP header field with details documented in [Authentication via Remote User Token](#)(see page 28).

#### User Token Realm

This realm activates token-based authentication for users as a substitute for plain-text username and password authentication. When the user token capability is enabled, the realm is automatically added to the Active Realms list. A full description of this realm is documented in [Accessing User Tokens in Realms](#)(see page 27).

❗ Removing all realms from the Active section prevents access to the repository manager for any user including any administrative access and has to be avoided.

## Privileges

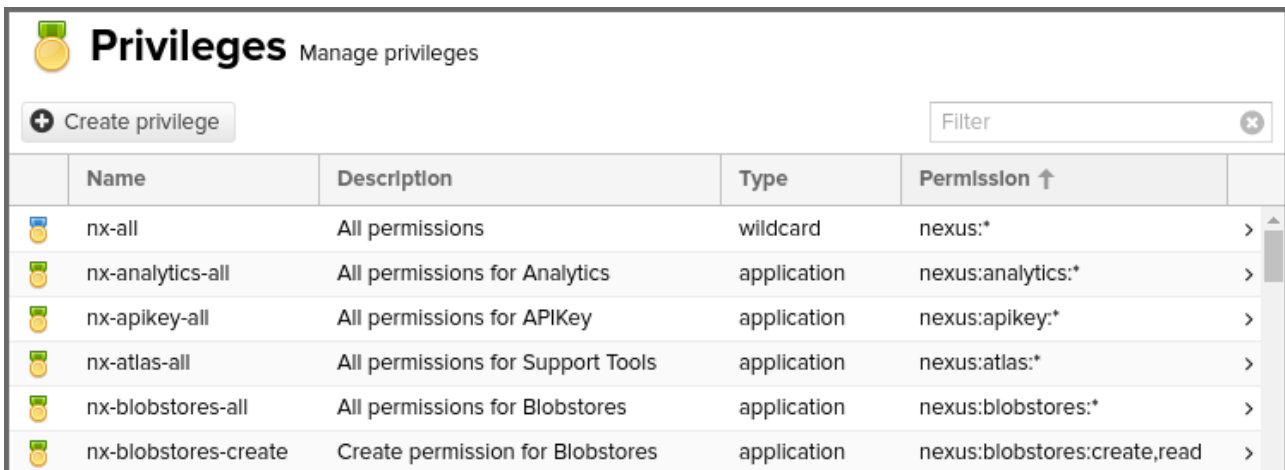
📘 Available in Nexus Repository OSS and Nexus Repository Pro







Privileges control access to specific functionality of the repository manager and can be grouped as a role and assigned to a specific users.

To access *Privileges* go to *Security* in the *Administration* menu, where it's listed as a sub-section. An extensive list of privileges is already built in the repository manager and is partially depicted in *Figure 6.2, "Partial List of Security Privileges"*. This feature allows you inspect existing privileges and create custom privileges.

<sup>3</sup> <https://help.sonatype.com/display/NXRM3/Node+Packaged+Modules+and+npm+Registries#NodePackagedModulesandnpmRegistries-npmSecurity>

<sup>4</sup> <https://help.sonatype.com/display/NXRM3/.NET+Package+Repositories+with+NuGet>



	Name	Description	Type	Permission ↑	
	nx-all	All permissions	wildcard	nexus:*	>
	nx-analytics-all	All permissions for Analytics	application	nexus:analytics:*	>
	nx-apikey-all	All permissions for APIKey	application	nexus:apikey:*	>
	nx-atlas-all	All permissions for Support Tools	application	nexus:atlas:*	>
	nx-blobstores-all	All permissions for Blobstores	application	nexus:blobstores:*	>
	nx-blobstores-create	Create permission for Blobstores	application	nexus:blobstores:create,read	>

**Figure 6.2. Partial List of Security Privileges**

The list of privileges displays an icon for the privilege *Type* as the first column, followed by:

#### **Name**

the internal identifier for the privilege

#### **Description**

a human readable description of the purpose of the privilege

#### **Type**

the aspect of the repository manager to which this privilege applies

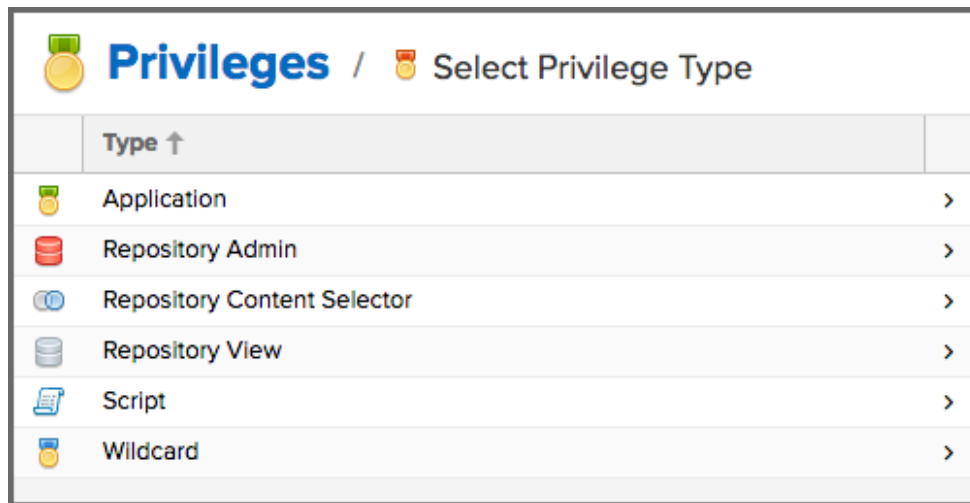
#### **Permission**

the internal permission definition as used by the embedded security framework

## Privilege Types

Further details are available after pressing on a specific row in the detail view.

Click the Create privilege button to view a list of privilege types, as seen in *Figure 6.3, “Choosing Privilege Types”*.



**Figure 6.3. Choosing Privilege Types**

Select the type corresponding to the area of the repository manager you wish to grant permissions. When you create a new *Privilege Type* you must assign at least one action in the *Actions* field.

The list of *Privilege Types* are as follows:

#### **Application**

These are privileges related to a specific domain in the repository manager

#### **Repository Admin**

These are privileges related to the administration and configuration of a specific repository

#### **Repository Content Selector**

These are privileges attributed to filtered content within a format, evaluated against a [content selector](#)<sup>5</sup>

#### **Repository View**

These are privileges controlling access to the content of a specific repository

#### **Script**

These are privileges related to the execution and management of scripts as documented in [REST and Integration API](#)<sup>6</sup>

#### **Wildcard**

These are privileges that use patterns to group other privileges

<sup>5</sup> <https://help.sonatype.com/display/NXRM3/Repository+Management>

<sup>6</sup> <https://help.sonatype.com/display/NXRM3/REST+and+Integration+API>



## Privilege Actions

Actions are functions allowing an explicit behavior the privilege can perform with the associated function.

The *Actions* to choose from are `add`, `browse`, `create`, `delete`, `edit`, `read`, `update`, and `*`. You can assign a single or combination of comma-delimited actions when creating new privileges. The privilege type to which you apply any of these Actions will perform the action's implied behavior. Consider how each action behaves when applied to a privilege type:

### `add`

This action allows privileges to add repositories or scripts.

### `browse`

This action allows privileges to view the contents of associated repositories. Unlike `read`, privilege types with `browse` can only view and administrate repository contents from UI.

### `create`

This action allows privileges to create applicable configurations within the repository manager. Since a `read` permission is required to view a configuration, this action is associated with most existing `create` privileges.

### `delete`

This action allows privileges to delete repository manager configurations, repository contents, and scripts. A `read` action is generally associated with `delete` actions so the actor can view these configurations to remove them.

### `edit`

This action allows privileges to modify associated scripts, repository content, and repository administration.

### `read`

This action allows privileges to view various configuration lists and scripts. Without `read`, any associated action will permit a privilege to see these lists but not its contents. The `read` action also allows privileges to utilize tools that can look at content from the command line.

### `update`

This action allows privileges to update repository manager configurations. Most existing privileges with `update` include `read` actions. Therefore, if creating custom privileges with `update`, the actor should consider adding `read` to the privilege in order to view repository manager configuration updates.

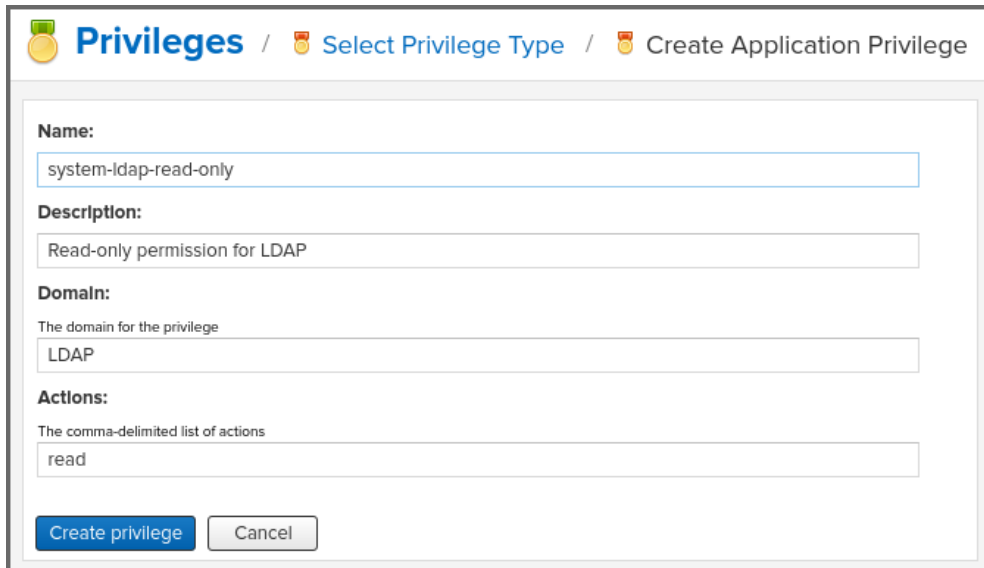
### `*`

This action is a wildcard giving you the ability to group all actions together.

To save a new custom privilege click the *Create privilege* button. The privilege can be found listed among the default privileges on the main *Privileges* screen. You can use the Filter input box to find a specific privilege.

## Example: Creating an Application Privilege

In the following example, an *Application* privilege type is created.



The screenshot shows a web form for creating a new application privilege. The breadcrumb navigation at the top reads "Privileges / Select Privilege Type / Create Application Privilege". The form has the following fields:

- Name:** A text input field containing "system-ldap-read-only".
- Description:** A text input field containing "Read-only permission for LDAP".
- Domain:** A text input field containing "LDAP". A small label "The domain for the privilege" is positioned above the input.
- Actions:** A text input field containing "read". A small label "The comma-delimited list of actions" is positioned above the input.

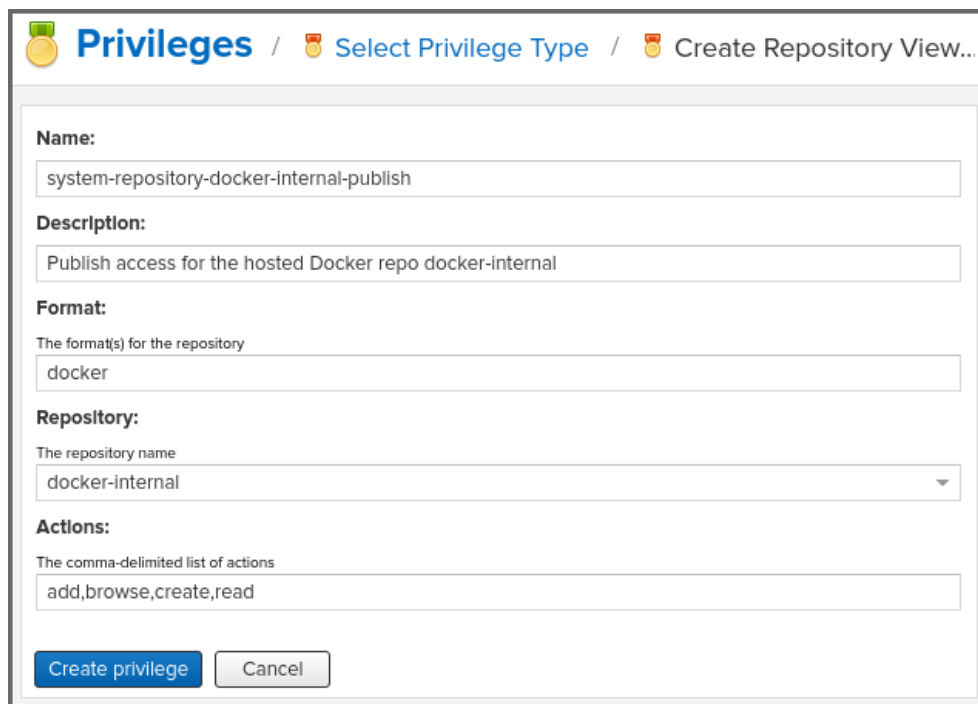
At the bottom of the form are two buttons: "Create privilege" (in blue) and "Cancel" (in grey).

**Figure 6.4. Creating an Application Privilege**

The form provides *Name*, *Description*, *Domain*, and *Actions*. In Figure 6.4, "Creating an Application Privilege" the form is completed for a privilege only that allows read access to the LDAP administration. If assigned this privilege, a user is able to view LDAP administration configuration but not edit it, create a new LDAP configuration, nor delete any existing LDAP configurations.

## Example: Creating a Repository View Privilege

In another example, a *Repository View* privilege type is created.



**Figure 6.5. Creating a Repository View Privilege**

The form provides *Name*, *Description*, *Format*, *Repository*, and *Actions*. In Figure 6.5, “Creating a Repository View Privilege” the form is completed for a privilege granting sufficient access to publish images to a specific hosted repository. A user with this privilege can view and read the contents of the repository as well as publish new images to it, but not delete images.

You can also assign privileges to users, and any assigned role, so they can have read-only access to a specific group repository. By default, these permissions will only allow users to read contents via the assigned group.

Additionally, users cannot access the contents of a group repository via members inside the group unless the member repository is assigned the same privileges as the group.

## Roles

 **Available in Nexus Repository OSS and Nexus Repository Pro**

Roles aggregate privileges into a related context and can, in turn, be grouped to create more complex roles.

The repository manager ships with a predefined admin as well as an anonymous role. These can be inspected in the *Roles* feature view accessible via the *Roles* item in the *Security* section of the Administration main menu. A simple example is shown in Figure 6.6, “Viewing the List of Defined Roles”. The list displays the Name and Description of the role as well as the Source, which displays whether the role is internal (Nexus) or a mapping to an external source like LDAP.



Roles Manage roles			
+ Create role ▾		Filter <span>✕</span>	
	Name	Source	Description ↑
	admin	Nexus	admin > 
	anonymous	Nexus	anonymous > 

**Figure 6.6. Viewing the List of Defined Roles**

To create a new role, click on the *Create role* button, select *Nexus Role* and fill out the *Role* creation feature view shown in *Figure 6.7, “Creating a New Role”*.

**Roles** / **Create Role**

**Role ID:**  
developer

**Role name:**  
Developer

**Role description:**  
Any software developer in our organization

**Privileges:**

Available

Filter

- nx-all
- nx-analytics-all
- nx-apikey-all
- nx-atlas-all
- nx-blobstores-all
- nx-blobstores-create

Given

**Roles:**

Available

Filter

- nx-admin

Contained

- nx-anonymous

**Create role** **Cancel**

**Figure 6.7. Creating a New Role**

When creating a new role, you will need to supply a *Role ID* and a Name and optionally a *Description*. Roles are comprised of other roles and individual privileges. To assign a role or privilege to a role, drag and drop the desired privileges from the Available list to the Given list under the *Privileges* header. You can use the *Filter* input to narrow down the list of displayed privileges and the arrow buttons to add or remove privileges.

The same functionality is available under the *Roles* header to select among the *Available* roles and add them to the list of *Contained* roles.

Finally press the *Create Role* button to get the role created.

An existing role can be inspected and edited by clicking on the row in the list. This role-specific view allows you to delete the role with the *Delete role* button. The built-in roles are managed by the repository manager and cannot be edited or deleted. The Settings section displays the same section as the creation view as displayed in *Figure 6.7, "Creating a New Role"*.

## Mapping External Groups to Nexus Roles

In addition to creating an internal role, the *Create role* button allows you to create an *External role mapping* to an external authorization system configured in the repository manager such as *LDAP*. This is something you would do, if you want to grant every member of an externally managed group (such as an *LDAP* group) a number of privileges and roles in the repository manager.

For example, assume that you have a group in *LDAP* named *scm* and you want to make sure that everyone in the *scm* group has administrative privileges.

Select *External Role Mapping* and *LDAP* to see a list of roles managed by that external realm in a dialog. Pick the desired *scm group* and confirm by pressing *Create mapping*.

**i** For faster access or if you cannot see your group name, you can also type in a portion or the whole name of the group and it will limit the dropdown to the selected text.

Once the external role has been selected, creates a linked role. You can then assign other roles and privileges to this new externally mapped role like you would do for any other role.

Any user that is part of the *scm* group in *LDAP*, receives all the privileges defined in the created role allowing you to adapt your generic role in *LDAP* to the repository manager-specific use cases you want these users to be allowed to perform.

## Users

**i** Available in Nexus Repository OSS and Nexus Repository Pro

The repository manager ships with two users by default: *admin* and *anonymous*. The *admin* user has all privileges and the *anonymous* user has read-only privileges. The default password for the *admin* user is *admin123*. The Users feature view displayed in *Figure 6.8, "Feature View with List of Users"* can be accessed via the *Users* item in the *Security* section of the *Administration* menu. The list shows the users *User ID*, *First*

*Name, Last Name* and *Email* as well as what security *Realm* is selected and if the accounts *Status* is *active* or *disabled*. The *Default* security realm is the local NXRМ realm.

	User ID ↑	Realm	First name	Last name	Email	Status	
	admin	default	Administrator	User	admin@example.o...	active	>
	anonymous	default	Anonymous	User	anonymous@exa...	active	>

**Figure 6.8. Feature View with List of Users**

Clicking on a user in the list or clicking on the *Create user* button displays the details view to edit or create the account shown in *Figure 6.9, "Creating or Editing a User"*. For external users, such as LDAP or Crowd, once you have your external realm setup you can edit their permissions here as well. Simply select the realm the user is on from the *Source* dropdown. Then type the *user ID* into the field to the right of that dropdown and search for it. Then click on the result desired to edit, same as a local user.

The screenshot shows the 'Users' management interface in Nexus Repository Manager 3. The breadcrumb navigation is 'Users / Administrator User'. Below the navigation, there are buttons for 'Delete user' and a 'More' dropdown menu. A 'Settings' tab is active. The form contains the following fields:

- ID:** A text field with the value 'admin'. A note below it says 'This will be used as the username'.
- First name:** A text field with the value 'Administrator'.
- Last name:** A text field with the value 'User'.
- Email:** A text field with the value 'admin@example.org'. A note below it says 'Used for notifications'.
- Status:** A dropdown menu currently set to 'Active'.
- Roles:** A section with two columns: 'Available' and 'Granted'.
  - Available:** Contains a 'Filter' input field and a list of roles: 'Developer' and 'nx-anonymous'.
  - Granted:** Contains a list of roles: 'nx-admin'.
  - Between the columns are two buttons: a right arrow (>) and a left arrow (<).

At the bottom of the form are two buttons: 'Save' and 'Discard'.

**Figure 6.9. Creating or Editing a User**

The ID can be defined upon initial creation and remains fixed thereafter. In addition you can specify the users *First Name*, *Last Name* and *Email address*. You also must enter and confirm a *Password*.

The *Status* allows you to set an account to be *Disabled* or *Active*. The *Roles* control allows you to add and remove defined roles to the user and therefore control the privileges assigned to the user. A user can be assigned one or more roles that in turn can include references to other roles or to individual privileges. For more information see [Roles](#)(see page 11).

On edit, the *More* button in the header allows you to select the *Change Password* item in the drop down. The password can be changed in a dialog, provided the user is managed by the built-in security realm.

For remote users, you can only edit, not create. Fields defined by the remote, such as ID, will be uneditable.



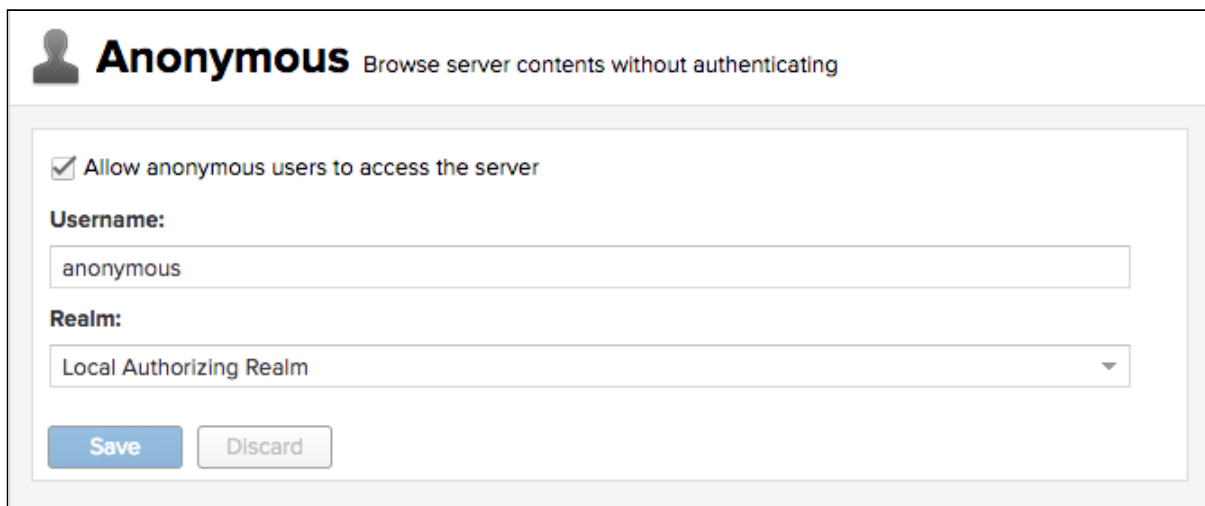
⚠ Ensure to change the password of the admin user to avoid security issues. Alternatively create other users with administrative rights and disable the default admin user.

## Anonymous Access

📘 Available in Nexus Repository OSS and Nexus Repository Pro

By default, the user interface as well as the repositories and the contained components are available to unauthenticated users for read access. The Anonymous feature view is available via the Anonymous item in the Security section of the Administration main menu and shown in *Figure 6.10, "Configuring Anonymous Access"*.

The privileges available to these users are controlled by the roles assigned to the anonymous user from the *NexusAuthorizingRole*. By changing the privileges assigned to this user in the [Users](#) (see page 14) feature view.



**Figure 6.10. Configuring Anonymous Access**

If you want to disable unauthenticated access to the repository manager entirely, you can uncheck the *Allow anonymous users to access the server* checkbox. The *Username* and *Realm* controls allow you to change the details for the anonymous user. E.g. you might have a guest account defined in your LDAP system and desire to use that user and the permissions it has for anonymous access.

## LDAP

 **Available in Nexus Repository OSS and Nexus Repository Pro**

Nexus Repository Manager can use the Lightweight Directory Access Protocol (LDAP) for authentication via external systems providing LDAP support such as Microsoft Exchange/Active Directory, OpenLDAP, ApacheDS and others.

Configuring LDAP can be achieved in a few simple steps:

- Enable LDAP Authentication Realm
- Create LDAP server configuration with connections and user/group mapping details
- Create external role mappings to adapt LDAP roles to repository manager specific usage

In addition to handling authentication, the repository manager can be configured to map roles to LDAP user groups. If a user is a member of a LDAP group that matches the ID of a role, the repository manager grants that user the matching role. In addition to this highly configurable user and group mapping capability, the repository manager can augment LDAP group membership with specific user-role mapping.

The repository manager can cache authentication information and supports multiple LDAP servers and user/group mappings. Connection details to the LDAP server and the user/group mappings as well as specific account logins can be tested directly from the user interface.

All these features allow you to adapt to any specific LDAP usage scenario and take advantage of the central authentication set up across your organization in all your repository managers.

## Enabling the LDAP Authentication Realm

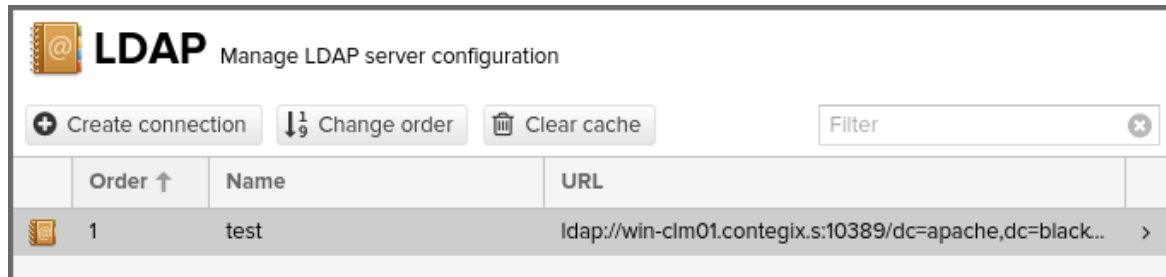
As shown in *Figure 6.1, "Security Realms Administration"*, activate your *LDAP Realm* by following these steps:

- Navigate to the *Realms* administration section
- Select the *LDAP Realm* and add it to the list of *Active* realms on the right
- Ensure that the *LDAP Realm* is located beneath the *Local Authenticating Realm* in the list
- Press *Save*

Best practice is to leave the *Local Authenticating Realm* and the *Local Authorizing Realm* activated so that the repository manager can be used by *anonymous*, *admin* and other users configured in this realm even with LDAP authentication offline or unavailable. Any user account not found in the *Local Authenticating Realm*, will be passed through to LDAP authentication.

## LDAP Connection and Authentication

The LDAP feature view, displayed in *Figure 6.11, “LDAP Feature View”*, is available via the *LDAP* item in the *Security* section of the *Administration* main menu.



**Figure 6.11. LDAP Feature View**

The *Order* determines in which order the repository manager connects to the LDAP servers when authenticating a user. The *Name* and *URL* columns identify the configuration and clicking on a individual row provides access to the *Connection* and *User and group* configuration sections.

The *Create connection* button can be used to create a new LDAP server configuration. Multiple configurations can be created and are accessible in the list.

The *Change order* button can be used to change the order in which the repository manager queries the LDAP servers in a pop up dialog.

Successful authentications are cached so that subsequent logins do not require a new query to the LDAP server each time. The *Clear cache* button can be used to remove these cached authentications.

- ✓ Contact the administrator of your LDAP server to figure out the correct parameters, as they vary between different LDAP server vendors, versions and individual configurations performed by the administrators.

The following parameters allow you to create an LDAP connection:

### Name

Enter a unique name for the new configuration.

### LDAP server address

Enter *Protocol*, *Hostname*, and *Port* of your LDAP server.

### Protocol

Valid values in this drop-down are `ldap` and `ldaps` that correspond to the Lightweight Directory Access Protocol and the Lightweight Directory Access Protocol over SSL.

### Hostname

The hostname or IP address of the LDAP server.

### Port

The port on which the LDAP server is listening. Port 389 is the default port for the `ldap` protocol, and port 636 is the default port for the `ldaps`.

### Search base

The search base further qualifies the connection to the LDAP server. The search base usually corresponds to the domain name of an organization. For example, the search base could be `dc=example,dc=com`.

*Note: If the values in your search base contain spaces, escape them with "%20", as in "dc=example%20corp,dc=com"*

You can configure one of four authentication methods to be used when connecting to the LDAP Server with the Authentication method drop-down.

### Simple Authentication

Simple authentication consists of a *Username* and *Password*. Simple authentication is not recommended for production deployments not using the secure `ldaps` protocol as it sends a clear-text password over the network.

### Anonymous Authentication

The anonymous authentication uses the server address and search base without further authentication.

### Digest-MD5

This is an improvement on the CRAM-MD5 authentication method. For more information, see RFC-2831.

### CRAM-MD5

The Challenge-Response Authentication Method (CRAM) is based on the HMAC-MD5 MAC algorithm. In this authentication method, the server sends a challenge string to the client. The client responds with a username followed by a Hex digest that the server compares to an expected value. For more information, see [RFC-2195](http://www.faqs.org/rfcs/rfc2195.html)<sup>7</sup>.

For a full discussion of LDAP authentication approaches, see [RFC-2829](http://www.ietf.org/rfc/rfc2829.txt/)<sup>8</sup> and RFC-2251.

### SASL Realm

The Simple Authentication and Security Layer (SASL) realm used to connect to the LDAP server. It is only available if the authentication method is Digest-MD5 or CRAM-MD5.

### Username or DN

*Username* or *DN* (Distinguished Name) of an LDAP user with read access to all necessary users and groups. It is used to connect to the LDAP server.

---

<sup>7</sup> <http://www.faqs.org/rfcs/rfc2195.html>

<sup>8</sup> <http://www.ietf.org/rfc/rfc2829.txt/>

## Password

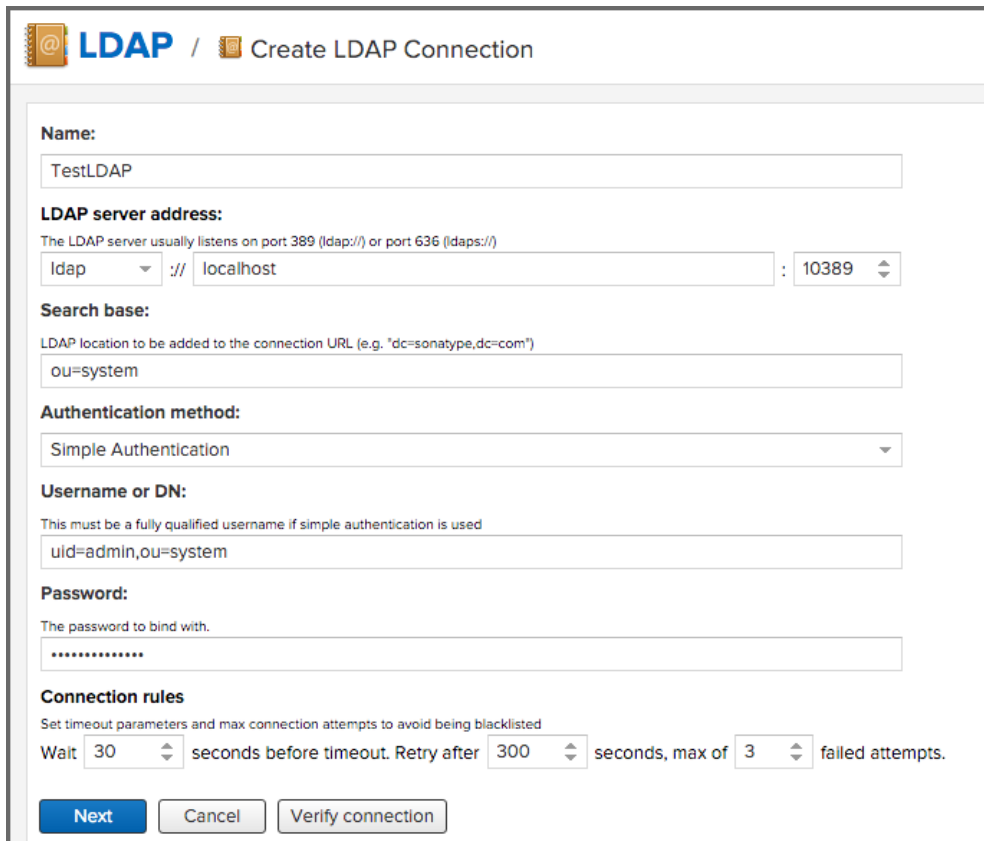
Password for the Username or DN configured above.

To test your connection to the external LDAP server, click *Verify connection*. A successful connection is confirmed with notification pop up.

The connection details can be further refined by configuring timeout period, retry period and number of connection attempts in *Connection rules*.

Click *Next* to proceed to configure user and group mappings for the LDAP configuration.

Figure 6.12, “Create LDAP Connection” shows a LDAP connection configuration for the repository manager configured to connect to an LDAP server running on localhost port 10389 using the search base of ou=system.



**LDAP** / Create LDAP Connection

**Name:**  
TestLDAP

**LDAP server address:**  
The LDAP server usually listens on port 389 (ldap://) or port 636 (ldaps://)  
ldap://localhost:10389

**Search base:**  
LDAP location to be added to the connection URL (e.g. "dc=sonatype,dc=com")  
ou=system

**Authentication method:**  
Simple Authentication

**Username or DN:**  
This must be a fully qualified username if simple authentication is used  
uid=admin,ou=system

**Password:**  
The password to bind with.  
\*\*\*\*\*

**Connection rules**  
Set timeout parameters and max connection attempts to avoid being blacklisted  
Wait 30 seconds before timeout. Retry after 300 seconds, max of 3 failed attempts.

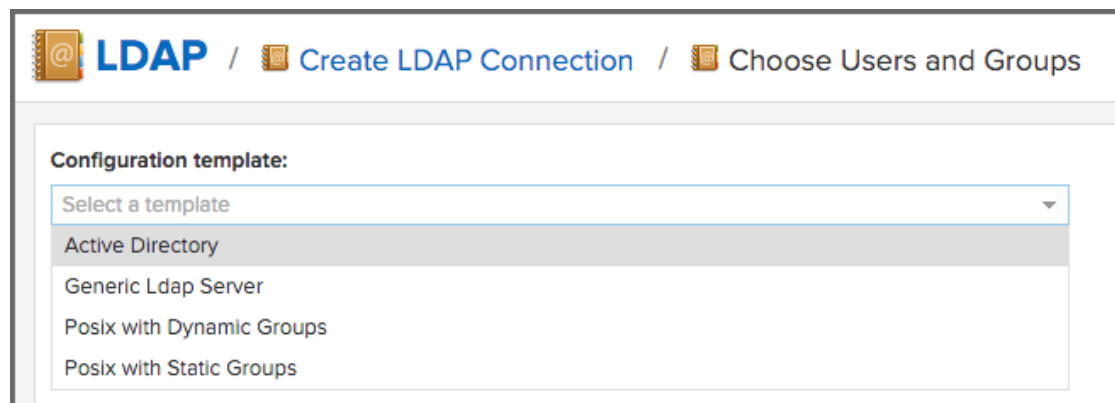
Next Cancel Verify connection

**Figure 6.12. Create LDAP Connection**

## User and Group Mapping

The LDAP connection panel contains a section to manage user and group mappings. This configuration is the next step after you configure and verify the LDAP Connection. It is separate panel called *Choose Users and Groups*.

This panel provides a *Configuration template* drop-down, shown in Figure 6.13, “*Configuration Template for Users and Groups*”. Based on your template selection the rest of the field inputs will adjust to the appropriate user and group template requirements. These templates are suggestions for typical configurations used on servers such as *Active Directory*, *Generic Ldap Server*, *Posix with Dynamic Groups*, and *Posix with Static Groups*. The values are suggestions only and have to be adjusted to your specific needs based on your LDAP server configuration.



**Figure 6.13. Configuration Template for Users and Groups**

The following parameters allow you to configure your user and group elements with the repository manager:

#### Base DN

Corresponds to the collection of distinguished names used as the base for user entries. This DN is relative to the Search Base. For example, if your users are all contained in `ou=users,dc=sonatype,dc=com` and you specified a Search Base of `dc=sonatype,dc=com`, you use a value of `ou=users`.

#### User subtree

Check the box if *True*. Uncheck if *False*. Values are true if there is a tree below the Base DN that can contain user entries and false if all users are contained within the specified Base DN. For example, if all users are in `ou=users,dc=sonatype,dc=com` this field should be *False*. If users can appear in organizational units within organizational units such as `ou=development,ou=users,dc=sonatype,dc=com`, this field should be *True*.

#### Object class

This value is a standard object class defined in RFC-2798. It specifies the object class for users. Common values are `inetOrgPerson`, `person`, `user`, or `posixAccount`.

#### User filter

This allows you to configure a filter to limit the search for user records. It can be used as a performance improvement.

#### User ID attribute

This is the attribute of the object class specified above, that supplies the identifier for the user from the LDAP server. The repository manager uses this attribute as the *User ID* value.

**Real name attribute**

This is the attribute of the Object class that supplies the real name of the user. The repository manager uses this attribute when it needs to display the real name of a user similar to usage of the internal *First name* and *Last name* attributes.

**Email attribute**

This is the attribute of the Object class that supplies the email address of the user. The repository manager uses this attribute for the *Email attribute* of the user. It is used for email notifications of the user.

**Password attribute**

It can be used to configure the Object class, which supplies the password ("userPassword"). If this field is blank the user will be authenticated against a bind with the LDAP server. The password attribute is optional. When not configured authentication will occur as a bind to the LDAP server. Otherwise this is the attribute of the Object class that supplies the password of the user. The repository manager uses this attribute when it is authenticating a user against an LDAP server.

An automatically checked box will allow you to Map LDAP groups as roles. With the configuration any LDAP group configured for a specific users is used to query the roles in the repository manager. Identical names trigger the user to be granted the privileges of the roles.

Groups in LDAP systems are configured to be dynamic or static. A dynamic group is a list of groups to which users belong. A static group contains a list of users. Select *Dynamic Groups* or *Static Groups* from the *Group type* drop-down to proceed with the appropriate configuration.

**Group type:**  
Static Groups

**Group base DN:**  
The base location in the LDAP that groups are found. This is relative to the search base (e.g. ou=Group).  
cn=builtin

**Group subtree:**  
☐ Are groups located in structures below the group base DN.

**Group object class:**  
LDAP class for group objects (e.g. posixGroup)  
group

**Group ID attribute:**  
sAMAccountName

**Group member attribute:**  
LDAP attribute containing the usernames for the group.  
member

**Group member format:**  
The format of user ID stored in the group member attribute (e.g. "uid=\${username},ou=people,o=sonatype")  
\${dn}

Save Discard Verify user mapping Verify login

**Figure 6.14. Static Group Element Mapping**

Static groups with an example displayed in *Figure 6.14, "Static Group Element Mapping"*, are configured with the following parameters:

#### Group Base DN

This field is similar to the Base DN field described for User Element Mapping, but applies to groups instead of users. For example, if your groups were defined under `ou=groups,dc=sonatype,dc=com`, this field would have a value of `ou=groups`.

#### Group subtree

This field is similar to the *User subtree* field described for User Element Mapping, but configures groups instead of users. If all groups are defined under the entry defined in Base DN, set the field to false. If a group can be defined in a tree of organizational units under the Base DN, set the field to true.

#### Group object class

This value in this field is a standard object class defined in RFC-2307. The class is simply a collection of references to unique entries in an LDAP directory and can be used to associate user entries with a group. Examples are `groupOfUniqueNames`, `posixGroup` or custom values.

#### Group ID attribute

Specifies the attribute of the object class that specifies the group identifier. If the value of this field corresponds to the ID of a role, members of this group will have the corresponding privileges.



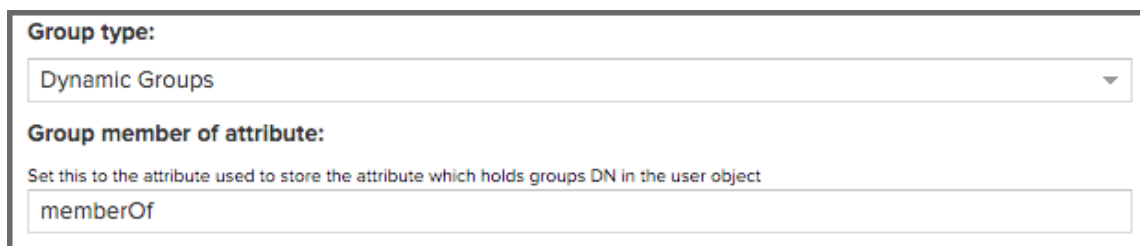
### Group member attribute

Specifies the attribute of the object class which specifies a member of a group. An example value is `uniqueMember`.

### Group member format

This field captures the format of the *Group Member Attribute*, and is used by the repository manager to extract a username from this attribute. An example values is `${dn}`.

If your installation does not use static groups, you can configure the LDAP connection to refer to an attribute on the user entry to derive group membership. To do this, select *Dynamic Groups* in the *Group type* drop down.



The screenshot shows a configuration form for dynamic groups. It has two main sections. The first section, titled 'Group type:', contains a dropdown menu with 'Dynamic Groups' selected. The second section, titled 'Group member of attribute:', contains a text input field with 'memberOf' entered. Below the text input field is a small, italicized instruction: 'Set this to the attribute used to store the attribute which holds groups DN in the user object'.

**Figure 6.15. Dynamic Group Element Mapping**

Dynamic groups are configured via the *Group member of attribute* parameter. The repository manager inspects this attribute of the user entry to get a list of groups of which the user is a member. In this configuration, seen in *Figure 6.15, "Dynamic Group Element Mapping"*, a user entry would have an attribute that would contain the name of a group, such as *memberOf*.

Once you have configured the user and group settings on the *Choose Users and Groups* form, you can check the correctness of your user mapping by pressing the *Verify user mapping* button. A successful mapping will result in the retrieval of a list of user records, which will be shown in the *User Mapping Test Result* dialog.

The repository manager provides you with the ability to test a user login directly. To test a user login, go to the *Choose Users and Groups* page after all appropriate field inputs of the form are filled. Scroll to the bottom and click the *Verify login* button.

The *Verify login* button can be used to check if authentication and user/group mappings work as expected for a specific user account besides the global account used for the LDAP configuration.

After the successful configuration of your LDAP connection and user and group mappings, you can proceed to configure external role mappings and assign them to users. This allows you to define the repository manager specific security for a LDAP group. More details are available in [Roles\(see page 11\)](#) and [Users\(see page 14\)](#).

## Security Setup with User Tokens

 **Available in Nexus Repository Pro only**

When using Apache Maven with Nexus Repository Manager Pro, the user credentials for accessing the repository manager have to be stored in the user's `settings.xml` file. Like a `pom.xml` your `settings.xml` is a file that contains your user preferences. The Maven framework has the ability to encrypt passwords within the `settings.xml`, but the need for it to be reversible in order to be used limits its security.

The default location of settings file is `~/.m2/settings.xml`. This file contains listings for personalized client or build-tool configurations such as repositories. This file is not exclusive to Maven-specific repositories.

Other build systems use similar approaches and can benefit from the usage of user tokens as well. Nexus Repository Manager Pro's user token feature establishes a two-part token for the user. Usage of the token acts as a substitute method for authentication that would normally require passing your username and password in plain text.


This is especially useful for scenarios where single sign-on solutions like LDAP are used for authentication against the repository manager and other systems and the plain text username and password cannot be stored in the `settings.xml` following security policies. In this scenario the generated user tokens can be used instead.

## Enabling and Resetting User Tokens

User token-based authentication can be activated by an administrator or user with the `nx-usertoken-settings` privilege. Users with that privilege must click the *User Token* menu item under *Security* in the *Administration* menu. Check the *Enable user tokens* box, then press *Save* to activate the feature.

Additionally, you can check the *Require user tokens for repository authentication* box to allow the repository manager to require a user token for any access to the repository and group content URLs. This affects read and write access for deployments from a build execution or a manual upload, but the user interface will not change.

You can also reset the token of an individual user by selecting the *User Token* tab in the *Users* administration from the *Security* menu. The password requested for the action to proceed is the password for the authenticated administrator who resets the token.

 **Resetting user tokens forces users to update the `settings.xml` with the newly created tokens, and could potentially break any command line builds using the tokens until this change is carried out. This also applies to continuous integration servers using user tokens or any other automated build executions.**

## Accessing User Tokens in Realms

When you activate user tokens, the feature automatically adds the *User Token Realm* to the *Active Realms* list. To see the results, go to *Realms* located under *Security* in the *Administration* menu. If desired, you can re-order the security realms used, although the default settings with the *User Token Realm* as the first realm is probably the desired setup. This realm is not removed when the user tokens are disabled; however, it will cleanly pass through to the next realm. The realm will remain in the active bin in your *Realms* in case the feature is reactivated at a later stage.

## Accessing and Using Your User Token

To grant users the ability to access user tokens:

1. Select *Roles* from *Security* in the *Administration* menu.
2. Choose a role you want to assign the permission, from the selection panel.
3. Assign the `nx-usertoken-current` privilege to the role, then save the change.

When enabled, the user can access their individual token from the mode toggle. To access the menu select the username, on the top right area of the main toolbar. In the User menu, to the left, the user will see the User Token menu item.

In order to see the *User Token* click the *Access user token* button. This will prompt the *Authenticate* dialog where you are required to re-enter your credentials. After clicking *Authenticate* in the completed dialog, another dialog will appear with the user token.

Below the *Access your token* section is another section that allows you to reset your token. Click the *Reset user token* button, which prompts an *Authenticate* dialog. Enter your credentials to complete the user token reset. Resetting the token will show a dialog with a success message, but you must access the user token again to see the new value.

The User Token dialog displays user code and pass code tokens in separate fields. Below the token, is the server section of your `settings.xml`. When using the server section you can replace the `{server}` placeholder with the repository id that references your repository manager you want to authenticate against with the user token. The dialog will close automatically after one minute or simply click the *Close* button.

The user code and pass code values can be used as replacements for username and password in the login dialog. You can still to use the original username and the pass code to log in to the user interface.

In order to utilize your user tokens for repository authentication you must access the repository manager with the user token, from the command line. For example, your username-password credentials access with:

```
curl -v --user admin:admin123 http://localhost:2468/repository/bower-all/
```

Or, you can replace those credentials with a user and pass code separated by a colon in the curl command line like this:

```
curl -v --user N+ZBiTlF:76xSi+HAQvYH2H8kgyJldwD7aJnPgCrHG/Zu7mkpWmZZ http://localhost:2468/repository/  
bower-all/
```

## Authentication via Remote User Token

**Available in Nexus Repository OSS and Nexus Repository Pro**

The repository manager allows integration with external security systems that can pass along authentication of a user via the `Remote_User` HTTP header field for all requests - Remote User Token authentication. This typically affects all web application usage in a web browser.

These are either web-based container or server-level authentication systems like [Shibboleth](#)<sup>9</sup>. In many cases, this is achieved via a server like [Apache HTTPD](#)<sup>10</sup> or [nginx](#)<sup>11</sup> proxying the repository manager. These servers can in turn defer to other authentication storage systems e.g., via the [Kerberos](#)<sup>12</sup> network authentication protocol. These systems and setups can be described as Central Authentication Systems CAS or Single Sign On SSO.

From the users perspective, he/she is required to login into the environment in a central login page that then propagates the login status via HTTP headers. the repository manager simply receives the fact that a specific user is logged in by receiving the username in a HTTP header field.

The HTTP header integration can be activated by adding and enabling the *Rut Auth* capability as documented in [Accessing and Configuring Capabilities](#)<sup>13</sup> and setting the *HTTP Header* name to the header populated by your security system. Typically, this value is `REMOTE_USER`, but any arbitrary value can be set. An enabled capability automatically causes the *Rut Auth Realm* to be added to the Active realms in the *Realms* configuration described in [Realms](#)(see page 4).

When an external system passes a value through the header, authentication will be granted and the value will be used as the user name for configured authorization scheme. For example, on a default installation with the internal authorization scheme enabled, a value of `admin` would grant the user the access rights in the user interface as the *admin* user.

---

<sup>9</sup> <http://shibboleth.net/>

<sup>10</sup> <http://httpd.apache.org/>

<sup>11</sup> <http://nginx.org/>

<sup>12</sup> <http://web.mit.edu/kerberos/>

<sup>13</sup> <https://help.sonatype.com/display/NXRM3/System+Configuration#SystemConfiguration-AccessingandConfiguringCapabilities>

A seamless integration can be set up for users if the external security system is exposed via LDAP and configured in the repository manager as LDAP authorization realm combined with external role mappings and in parallel the sign-on is integrated with the operating system sign-on for the user.

## Configuring SSL

Using Secure Socket Layer (SSL) communication with the repository manager is an important security feature and a recommended best practice. Secure communication can be inbound or outbound.

Outbound client communication may include integration with:

- a remote proxy repository over HTTPS - documented in [Repository Management](#)<sup>14</sup>
- SSL/TLS secured servers - e.g. for SMTP/email integration documented in [Email Server](#)<sup>15</sup>
- LDAP servers configured to use LDAPS
- specialized authentication realms such as the [Crowd](#)<sup>16</sup> realm.

Inbound client communication includes

- web browser HTTPS access to the user interface,
- tool access to repository content,
- and manual or scripted usage of the REST APIs.

## Outbound SSL - Trusting SSL Certificates of Remote Repositories

**Available in Nexus Repository OSS and Nexus Repository Pro**

When the SSL certificate of a remote proxy repository is not trusted, the repository may be automatically blocked outbound requests fail with a message similar to "PKIX path building failed".

The Proxy configuration for each proxy repository documented in [Managing Repositories and Repository Groups](#)<sup>17</sup> includes a section titled *Use the Nexus truststore*. It allows you to manage the SSL certificate of the remote repository and solves these problems. It is only displayed, if the remote storage uses a HTTPS URL.

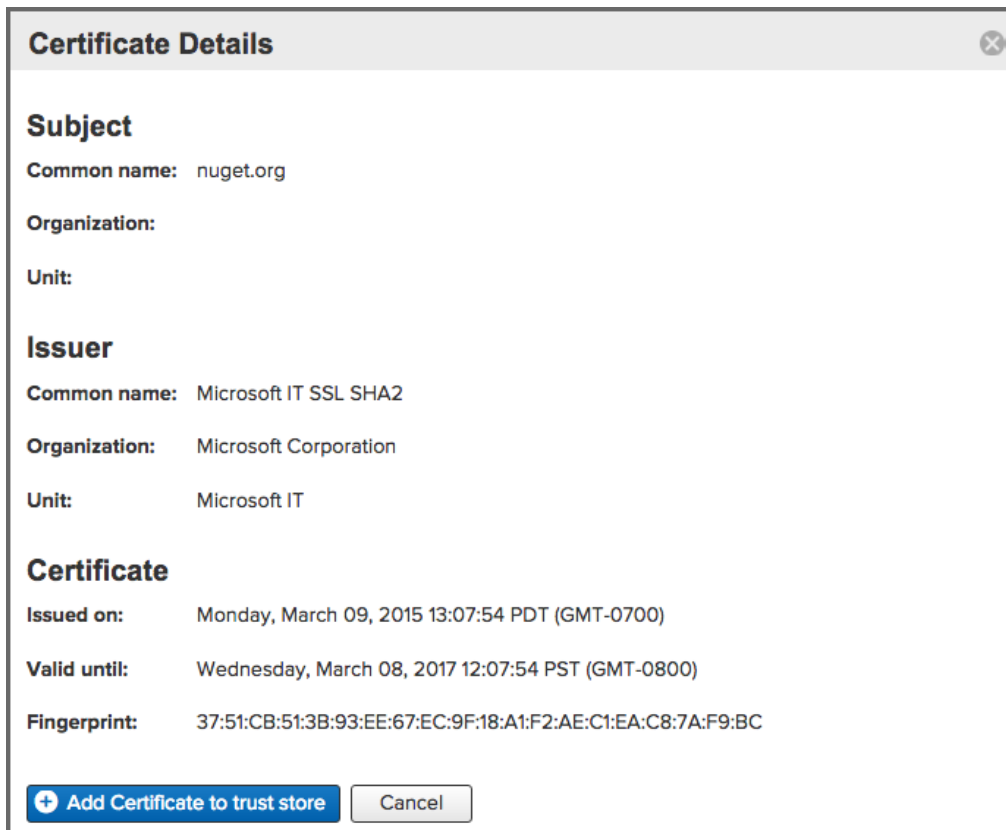
The *View certificate* button triggers the display of the SSL Certificate Details dialog. An example is shown in *Figure 6.16, "Certificate Details Dialog to Add an SSL to the Nexus Truststore"*.

<sup>14</sup> <https://help.sonatype.com/display/NXRM3/Repository+Management>

<sup>15</sup> <https://help.sonatype.com/display/NXRM3/System+Configuration#SystemConfiguration-Email/SMTPServerConfiguration>

<sup>16</sup> <https://help.sonatype.com/display/NXRM3/Atlassian+Crowd+Support>

<sup>17</sup> <https://help.sonatype.com/display/NXRM3/Repository+Management#RepositoryManagement-ManagingRepositoriesandRepositoryGroups>



**Figure 6.16. Certificate Details Dialog to Add an SSL to the Nexus Truststore**

Use the *Certificate Details* dialog when the remote certificate is not issued by a well-known public certificate authority included in the default Java trust store. This specifically also includes usage of self-signed certificates used in your organization. To confirm trust of the remote certificate, click the *Add certificate to truststore* button in the dialog. This feature is analogous to going to the SSL Certificates user interface and using the Load certificate button found there as described in [Outbound SSL - Trusting SSL Certificates Globally](#) (see page 31). If the certificate is already added, the button can undo this operation and will read *Remove certificate from trust store*.

The checkbox labelled Use certificates stored in Nexus to connect to external systems is used to confirm that the repository manager should consult the internal truststore as well as the JVM truststore when confirming trust of the remote repository certificate. Without adding the certificate to the private truststore and enabling the checkbox, the repository will not trust the remote.

The default JVM truststore of the JVM installation used to run the repository manager and the private truststores are merged. The result of this merge is used to decide about the trust of the remote server. The default Java truststore already contains public certificate authority trust certificates. If the remote certificate is signed by one of these authorities, then explicitly trusting the remote certificate will not be needed.

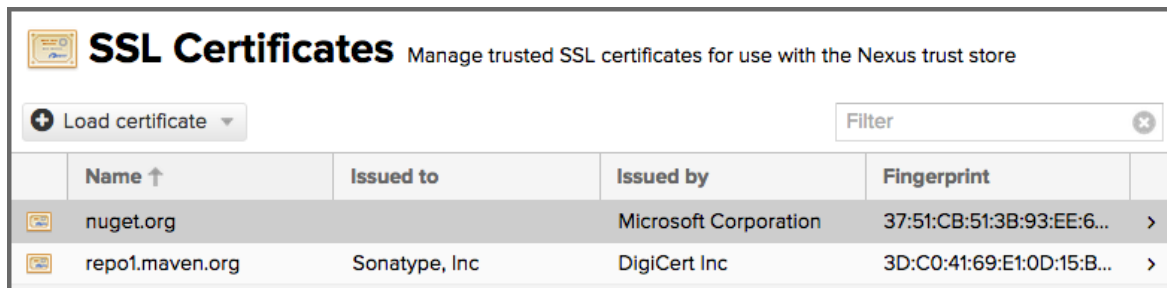
⚠ When removing a remote trusted certificate from the truststore, a repository manager restart is required before a repository may become untrusted.

## Outbound SSL - Trusting SSL Certificates Globally

### Available in Nexus Repository OSS and Nexus Repository Pro

The repository manager allows you to manage trust of all remote SSL certificates in a centralized user interface. Use this interface when you wish to examine all the currently trusted certificates for remote repositories, or manage certificates from secure remotes that are not repositories.

Access the feature view for SSL Certificates administration by selecting the *SSL Certificates* menu items in the *Security* sub-menu in the *Administration* main menu.



Name ↑	Issued to	Issued by	Fingerprint
nuget.org		Microsoft Corporation	37:51:CB:51:3B:93:EE:6...
repo1.maven.org	Sonatype, Inc	DigiCert Inc	3D:C0:41:69:E1:0D:15:B...

**Figure 6.17. SSL Certificates Administration**

The list shows any certificates that are already trusted. Clicking on an individual row allows you to inspect the certificate. This detail view shows further information about the certificate including *Subject*, *Issuer*, and *Certificate* details. The *Delete certificate* button allows you to remove a certificate from the truststore.

The button *Load certificate* above the list of certificates can be used to add a new certificate to the truststore by loading it directly from a server or using a PEM file representing the certificate.

The common approach is to choose *Load from server* and enter the full `https://` URL of the remote site, e.g, `https://repo1.maven.org`. The repository manager will connect using HTTPS and use the HTTP proxy server settings if applicable. When the remote is not accessible using `https://`, only enter the host name or IP address, optionally followed by colon and the port number. For example: `example.com:8443`. In this case the repository manager will attempt a direct SSL socket connection to the remote host at the specified port. This allows you to load certificates from SMTP or LDAP servers, if you use the correct port.

Alternatively you can choose the *Paste PEM* option to configure trust of a remote certificate. Copy and paste the Base64 encoded X.509 DER certificate to trust. This text must be enclosed between lines containing `-----BEGIN CERTIFICATE-----`

and `-----END CERTIFICATE-----`.

Typically this file is supplied to you by the certificate owner. An example method to get the encoded X.509 certificate into a file on the command line using `keytool` is:

```
keytool -printcert -rfc -sslserver repo1.maven.org > repo1.pem
```

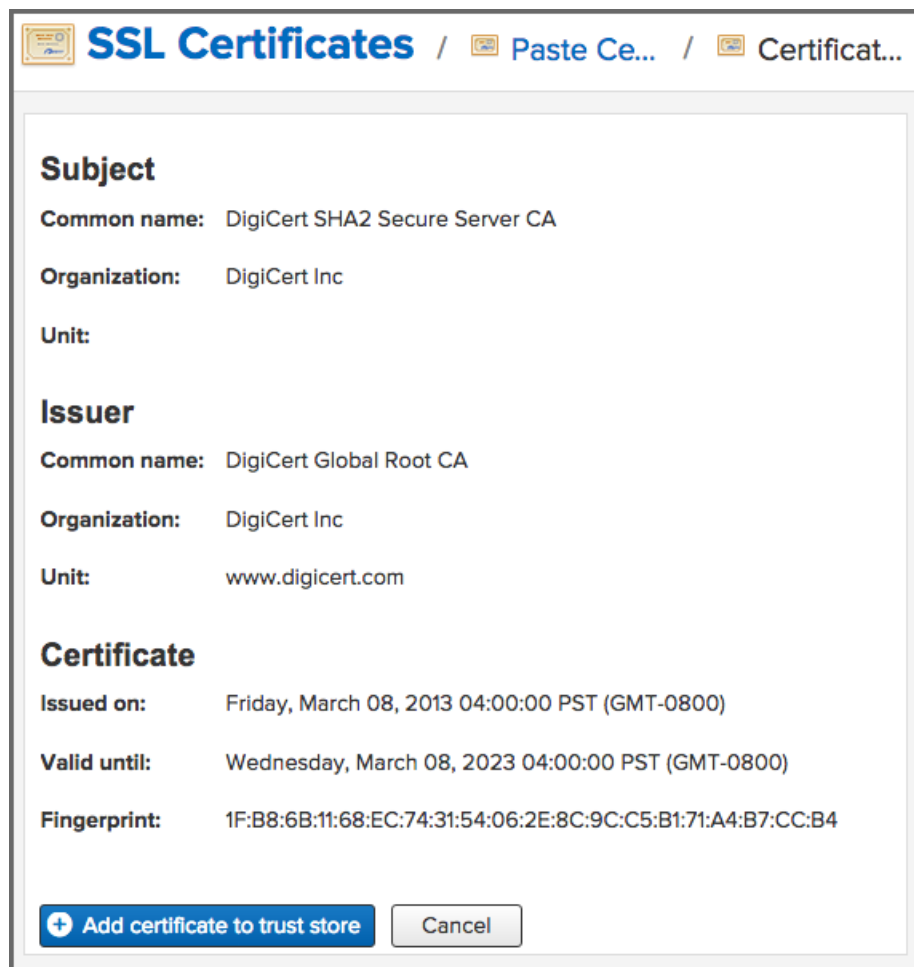
The resulting `repo1.pem` file contains the encoded certificate text that you can cut and paste into the dialog in the user interface. An example of inserting such a certificate is shown in *Figure 6.18, "Providing a Certificate in PEM Format"*.



**Figure 6.18. Providing a Certificate in PEM Format**

If the repository manager can successfully retrieve the remote certificate or decode the pasted certificate, the details will be shown allowing you to confirm details as shown in *Figure 6.19, "Certificate Details Displayed after Successful Retrieval or Parsing"*. Please review the displayed information carefully before clicking *Add Certificate* to establish the truststore addition.





**Figure 6.19. Certificate Details Displayed after Successful Retrieval or Parsing**

In some organizations, all of the remote sites are accessed through a globally configured proxy server which rewrites every SSL certificate. This single proxy server is acting as a private certificate authority. In this case, you can follow special instructions for trusting the proxy server root certificate, which can greatly simplify your certificate management duties.

## Outbound SSL - Trusting SSL Certificates Using Keytool

### Available in Nexus Repository OSS and Nexus Repository Pro

Managing trusted SSL certificates from the command line using `keytool`<sup>18</sup> and system properties is an alternative and more complex option than using the SSL certificate management features of the repository manager.

Before you begin the process of trusting a certificate from the command line you will need:

- a basic understanding of SSL certificate technology and how the Java VM implements this feature

---

<sup>18</sup> <http://docs.oracle.com/javase/8/docs/technotes/tools/index.html#security>

- command line access to the host operating system and the `keytool` program
- network access to the remote SSL server you want to trust from the host running the repository manager. This must include any HTTP proxy server connection details.

If you are connecting to servers that have certificates which are not signed by a public CA, you will need to complete these steps:

1. Copy the default JVM truststore file `$JAVA_HOME/jre/lib/security/cacerts` to `$data-dir/custom-truststore.jks` for editing. Ensure the file permissions allow read for the repository manager user.
2. Import additional trusted certificates into the copied truststore file.
3. Configure JSSE system properties for the repository manager process so that the custom truststore is consulted instead of the default file.

Some common commands to manually trust remote certificates can be found in our [SSL Certificate Guide](#)<sup>19</sup>.

After you have imported your trusted certificates into a truststore file, you can add the JVM parameters configuring the truststore file location and password as separate configuration lines into the file `$install-dir/bin/nexus.vmoptions`

```
-Djavax.net.ssl.trustStore=<absolute_path_to_custom_truststore_file>
-Djavax.net.ssl.trustStorePassword=<truststore_password>
```

Once you have added the properties shown above, restart the repository manager and attempt to proxy a remote repository using the imported certificate. The repository manager will automatically register the certificates in the truststore file as trusted.

## Inbound SSL - Configuring to Serve Content via HTTPS

### Available in Nexus Repository OSS and Nexus Repository Pro

Providing access to the user interface and content via HTTPS is a best practice.

You have two options:

- Use a separate reverse proxy server in front of the repository manager to manage HTTPS
- Configure the repository manager itself to serve HTTPS directly

## Using A Reverse Proxy Server

A common approach is to access the repository manager through a dedicated server which answers HTTPS requests on behalf of the repository manager - these servers are called reverse proxies or SSL/TLS

---

<sup>19</sup> <https://support.sonatype.com/hc/en-us/articles/213465768-SSL-Certificate-Guide#common-keytool-commands>

terminators. Subsequently requests are forwarded to the repository manager via HTTP and responses received via HTTP are then sent back to the requestor via HTTPS.

There are a few advantages to using these which can be discussed with your networking team. For example, the repository manager can be upgraded/installed without the need to work with a custom JVM keystore. The reverse proxy could already be in place for other systems in your network. Common reverse proxy choices are Apache httpd, nginx, Eclipse Jetty or even dedicated hardware appliances. All of them can be configured to serve SSL content, and there is a large amount of reference material available online.

## Serving SSL Directly

The second approach is to use the Eclipse Jetty instance that is distributed with the repository manager to accept HTTPS connections.

### How to Enable the HTTPS Connector

1. Create a Java keystore file at `$install-dir/etc/ssl/keystore.jks` which contains the Jetty SSL certificate to use. Instructions are available on the [Eclipse Jetty documentation](#)<sup>20</sup> site or our [SSL Certificate Guide](#)<sup>21</sup>.
2. Edit `$data-dir/etc/nexus.properties`. **Add** a property on a new line `application-port-ssl=8443`. Change 8443 to be your preferred port on which to expose the HTTPS connector. Do not choose a port already in use by another connector or process on the same host.
3. Edit `$data-dir/etc/nexus.properties`. Uncomment the line containing `nexus-args` by removing the leading `#` and space characters. Also edit the comma delimited property value to include `${jetty.etc}/jetty-https.xml`. Save the file. Example edited line with comment character removed and `${jetty.etc}/jetty-https.xml` added to the existing values:  
`nexus-args=${jetty.etc}/jetty.xml,${jetty.etc}/jetty-http.xml,${jetty.etc}/jetty-https.xml,${jetty.etc}/jetty-requestlog.xml`
4. Edit `$install-dir/etc/jetty/jetty-https.xml` and set your keystore passwords appropriately. Make sure all three passwords are the same value. Note that if you use "password" as your keystore password this step is not necessary.
5. Restart the repository manager. Verify HTTPS connections can be established.
6. Update the Base URL to use https in your repository manager configuration using the [Base URL](#)<sup>22</sup> capability.

<sup>20</sup> <https://www.eclipse.org/jetty/documentation/9.4.x/configuring-ssl.html>

<sup>21</sup> <https://support.sonatype.com/hc/en-us/articles/213465768-SSL-Certificate-Guide>

<sup>22</sup> <https://help.sonatype.com/display/NXRM3/System+Configuration#SystemConfiguration-BaseURLCreation>

**Tip:**

While the above works, it will require that you manually reconfigure SSL whenever you upgrade Nexus Repo Manager. Use the following steps to create an SSL setup that will survive upgrade.

Create your keystore in `$data-dir/etc/ssl/keystore.jks` with password of "password". By doing that you don't have to modify the default `jetty-https.xml` file NXRM ships with.

Then in `$data-dir/etc/nexus.properties` you'd add these lines:

```
nexus-args=${jetty.etc}/jetty.xml,${jetty.etc}/jetty-http.xml,${jetty.etc}/jetty-requestlog.xml,${jetty.etc}/jetty-https.xml
ssl.etc=${karaf.data}/etc/ssl
application-port-ssl=8443
```

## How to Redirect All Plain HTTP Requests to HTTPS

Some organizations need to remind their users that Nexus Repository Manager should only be used over HTTPS and redirecting HTTP requests to HTTPS can help. Do the following:

1. Follow all the steps under [How to Enable the HTTPS Connector](#)(see page 35). Make sure the `nexus-args` property value still includes the reference to `${jetty.etc}/jetty-http.xml`
2. Edit `$data-dir/etc/nexus.properties`. Change the `nexus-args` property comma delimited value to include `${jetty.etc}/jetty-http-redirect-to-https.xml`. Save the file.
3. Restart the repository manager. Verify all plain HTTP requests get redirected to the equivalent HTTPS URL.

**ⓘ** Redirecting HTTP requests is not recommended because it introduces implied security and creates increased network latency. Clients which send Basic Authorization headers preemptively may unintentionally expose credentials in plain text.

## How to Disable the HTTP Connector

1. Edit `$data-dir/etc/nexus.properties`. Change the `nexus-args` property comma delimited value to not include `${jetty.etc}/jetty-http.xml`. Save the file.
2. Restart the repository manager. Verify plain HTTP requests are no longer serviced.

## Auditing

Auditing of Nexus Repository Manager is done by enabling a capability called *Audit* as described in [Accessing and Configuring Capabilities](#)<sup>23</sup>. For your convenience, this capability is created by default in Nexus Repository Manager installations but is disabled.

Once enabled, a left navigation item *Audit* will appear under the *System* submenu of the *Administration* section. Clicking on this item reveals a table of audit records that have occurred in your Nexus Repository Manager instance, like the example shown in *Figure 6.20, “Example of Audit Table with Expanded Line Item”*. This table data persists through server restart but can be manually cleared using the *Clear* button above the table results.

	Domain	Type	Context	Timestamp ↑	Initiator
+	repository	updated	maven-central	2016-Nov-21 18:07:24	admin/127.0.0.1
+	security.privilege	created	all	2016-Nov-21 18:08:17	admin/127.0.0.1
+	security.privilege	deleted	all	2016-Nov-21 18:08:20	admin/127.0.0.1
-	security.role	created	testing	2016-Nov-21 18:08:35	admin/127.0.0.1
<b>Domain</b> security.role <b>Type</b> created <b>Context</b> testing <b>Timestamp</b> Monday, November 21, 2016 18:08:35 EST (GMT-0500) <b>Node ID</b> F3F827AA-B975FBF9-EAA1C218-D683B724-01D28AA8 <b>Initiator</b> admin/127.0.0.1 <b>Attribute: name</b> testing <b>Attribute: privileges</b> nx-analytics-all <b>Attribute: id</b> testing <b>Attribute: source</b> default <b>Attribute: roles</b>					
+	security.user	created	joedragons	2016-Nov-21 18:08:53	admin/127.0.0.1
+	security.anonymous	changed	system	2016-Nov-21 18:08:58	admin/127.0.0.1
+	security.anonymous	changed	system	2016-Nov-21 18:08:59	admin/127.0.0.1
+	security.idap	created	1a74401f-c104-4061-bbc7...	2016-Nov-21 18:09:21	admin/127.0.0.1
+	security.realm	changed	system	2016-Nov-21 18:09:21	admin/127.0.0.1
+	security.realm	changed	system	2016-Nov-21 18:09:26	admin/127.0.0.1
+	security.sslcertificate	created	www.google.com	2016-Nov-21 18:09:36	admin/127.0.0.1
+	capability	activated	analytics.collection	2016-Nov-21 18:09:41	admin/127.0.0.1

**Figure 6.20. Example of Audit Table with Expanded Line Item**

The table contains a record for every configuration change, as well as any asset or component additions and removals. Each row will give you some details about the event, including the type of event, when it happened and which user initiated the action. In addition to what is shown, each line item can be expanded to show more information by clicking the + sign at the beginning of the row. The content of the expanded information

<sup>23</sup> <https://help.sonatype.com/display/NXRM3/System+Configuration#SystemConfiguration-AccessingandConfiguringCapabilities>

varies slightly by the Domain viewed. You can collapse the additional information by clicking the – sign at the beginning of the row. The table displays about 250 rows and if there are more than that you need to use pagination to see more entries.